2

FINAL TECHNICAL REPORT

EXPENDABLE AIR VEHICLES/ HIGH ALTITUDE BALLOON TECHNOLOGY

2 August 1991

CHR/91-2750

Sponsored By:
Defense Advanced Research Projects Agency (DoD)
Defense Small Business Innovation Research Program
ARPA Order No. 6685

Issued By U.S. Army Missile Command
Under Contract Number:
DAAH01-90-C-0234

Prepared By:
Robert L. Hawkins, Principal Investigator
Coleman Research Corporation
6820 Moquin Drive
Huntsville, AL 35806

Contract Effective: 31 January 1990
Contract Expires: 2 August 1991

"The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either express or implied, of the Defense Advanced Research Projects Agency or the U.S. Government."

Coleman Research Corporation

HIGH TECHNOLOGY AEROSPACE SYSTEMS DESIGN

ORLANDO HUNTSVILLE LAUREL EL PASO WASHINGTON, D.C.

Coleman Research Corporation

HEADQUARTERS

5950 Lakehurst Drive
Orlando, FL 32819
(407) 352-3700
FAX: (407) 345-8616

ORLANDO DIVISION EAST

3045 Technology Parkway
Orlando, FL 32826-3299
(407) 249-7717
FAX: (407) 381-3980

HUNTSVILLE DIVISION

6820 Moquin Drive
Huntsville, AL 35806
(205) 922-6000
FAX: (205) 922-6053

LAUREL DIVISION

14502 Greenview Drive, Suite 206
Laurel, Maryland 20708
(301) 470-3839
FAX: (301) 776-5461

SOUTHWEST DIVISION

3 Butterfield Trail Blvd,
Suite 119
El Paso, TX 79906
(915) 772-4444
FAX: (915) 779-8754

WASHINGTON DIVISION

9302 Lee Highway,
Suite 800
Fairfax, VA 22031
(703) 934-7810
FAX: (703) 934-7800

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS
N/A

2a. SECURITY CLASSIFICATION AUTHORITY
N/A

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE
N/A

3. DISTRIBUTION / AVAILABILITY OF REPORT
Unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
CHR/91-2750

5. MONITORING ORGANIZATION REPORT NUMBER(S)
N/A

6a. NAME OF PERFORMING ORGANIZATION
Coleman Research Corp.

6b. OFFICE SYMBOL *(If applicable)*
N/A

7a. NAME OF MONITORING ORGANIZATION
DARPA
U.S. Army Missile Command

6c. ADDRESS *(City, State, and ZIP Code)*
6820 Moquin Drive
Huntsville, AL 35806

7b. ADDRESS *(City, State, and ZIP Code)*
AMSMI-RD-DP-TT
Redstone Arsenal, AL 35898-5244

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
DARPA

8b. OFFICE SYMBOL *(If applicable)*

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
Contract Number DAAH01-90-C-0234

8c. ADDRESS *(City, State, and ZIP Code)*
1400 Wilson Blvd.
Arlington, VA 22209-2308

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO.
N/A	N/A	N/A	N/A

11. TITLE *(Include Security Classification)*
Final Technical Report

12. PERSONAL AUTHOR(S)

13a. TYPE OF REPORT
Final Technical Report

13b. TIME COVERED
FROM 1/31/90 TO 8/2/91

14. DATE OF REPORT *(Year, Month, Day)*
1991 August 2

15. PAGE COUNT

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
N/A

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

The development of a digital computer simulation for high-altitude, scientific balloon drift pattern prediction is described. Provision is made for the use of either wind forecast data or a worldwide, empirical atmospheric data base. The program operates on a Macintosh computer and produces on-screen drift patterns which may be imported into commercially available graphics programs for annotation and/or printing. The report includes a listing of the FORTRAN source code for the program.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
□ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT □ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION

22a. NAME OF RESPONSIBLE INDIVIDUAL
Robert L. Hawkins, Principal Investigator

22b. TELEPHONE *(Include Area Code)*
(205) 922-6000

22c. OFFICE SYMBOL

Previous editions are obsolete.

BALLOON DRIFT PATTERN SIMULATION (BDPS)
FINAL TECHNICAL REPORT

2 August 1991

CHR/91-2750

Reporting Period:
31 January 1990 through 2 August 1991

In Accordance With Contract Number
DAAH01-90-C-0234 under CDRL A004

Prepared for:
U.S. Army Missile Command
DARPA Project Office
Redstone Arsenal, AL 35889-5280

Prepared by:
Coleman Research Corporation
6820 Moquin Drive
Huntsville, AL 35806

THIS PAGE INTENTIONALLY LEFT BLANK

PREFACE

The work described in this Phase II SBIR Final Technical Report is the implementation of a capability which Coleman Research Corporation demonstrated during a Phase I SBIR (contract number DAAH01-90-C-0234). Both contracts were performed under the administrative overview of the USAMICOM DARPA Project Office at Redstone Arsenal, Alabama. Technical guidance was given by DARPA from their Arlington, Virginia headquarters.

ACKNOWLEDGMENTS

Coleman Research Corporation wishes to acknowledge several people who have provided significant assistance during the execution of the work described in this contract.

Mr. Lawrence J. Schilling, NASA/Ames Research Center, Dryden Flight Research Facility: Mr. Schilling provided us with a "real-time" version of NASA's Global Reference Atmosphere Model (GRAM) which he had developed for use with National AeroSpace Plane (NASP) trajectory simulations.

Mr. Dale Johnson, NASA/Marshall Space Flight Center: Mr. Johnson, as the GRAM sponsor, provided us with AIAA articles and general GRAM information.

Mr. Joseph L. Lindinger, U.S. Navy/Naval Air Development Center: Mr. Lindinger arranged for us to have access to the Navy's atmospheric forecasting capability at the Fleet Numerical Oceanography Center.

Finally, Mr. Robert L. Hawkins, the Principal Investigator of this SBIR program, would like to acknowledge those individuals at CRC who contributed to this effort: Mr. Michael C. Bateman, Mr. David H. Wales, Mr. Norman E. Mason. Mr. David F. Smith, and Mr. A. Scott Cadenhead.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

1.0 INTRODUCTION

Under the sponsorship of the Defense Advanced Research Projects Agency (DARPA), Coleman Research Corporation (CRC) has developed a Balloon Drift Pattern Simulation (BDPS). CRC developed this simulation software for digital computers as a product of a Phase II Small Business Innovative Research (SBIR) project. This report presents the approach which CRC took to develop this product.

DARPA is interested in exploiting high-altitude, expendable balloon-borne communication and surveillance capabilities for military applications. Balloons offer several unique capabilities when high-altitude balloon technologies are integrated with payloads employing the power-efficient, light-weight electronic technologies available today. In order to establish communication and surveillance systems design requirements and deployment schedules, DARPA must analyze upper-atmosphere drift pattern simulation results for a variety of balloon payload concepts.

The Phase II development of BDPS was the logical activity to follow CRC's Phase I SBIR program [1] in which we demonstrated the technical feasibility of predicting high-altitude balloon drift patterns using a digital computer simulation. CRC realized the Phase I goal by completing each of four contract objectives: performing a literature survey, developing and integrating atmosphere and balloon models into a Balloon Drift Pattern Simulation, developing drift pattern output formats, and exercising the BDPS to produce and analyze balloon drift patterns. Drift patterns were produced for two balloon configurations which were specified by DARPA. The first mission employed a zero-pressure balloon which floated at an altitude of 70,000 feet over 24 hours. The second mission used a super-pressure balloon which floated at 120,000 feet for a one-year period. The models were executed and display output produced on VAX computers.

The project objective of the Phase II contract was to develop a Macintosh-based BDPS for DARPA. The accomplishment of this overall objective depended upon the completion of specific technical objectives, which are discussed in the following sections of this report: (1) Develop a suitable atmosphere model; (2) Streamline and modify the trajectory model; (3) Install BDPS on the Macintosh; (4) Develop Macintosh graphics output; and (5) Develop BDPS documentation

2.0 BDPS ATMOSPHERE MODEL

The first technical objective to be met was the development of a suitable atmosphere model for use within BDPS. Since BDPS was intended to be used as a deployment tool, the atmosphere model was of critical importance because the wind would have the most significant impact on balloon motion.

CRC decided to design the BDPS tool to be able to use either of two atmosphere mc · ls options. The first option uses wind tables which the user must provide. The second option makes use of an empirical climate model for which we have included climate data files with the delivered software.

2.1 WIND TABLE OPTION

This option was provided to give the BDPS user a mechanism by which wind data from a variety of sources may be incorporated to support drift pattern prediction. The option requires the user to supply two files in a format which is specified in the User Manual appendix. These two files contain, respectively, east wind velocity and north wind velocity. The velocity parameters in each file are organized as functions of latitude, longitude, and altitude.

This option has been specifically designed to support retrieval of forecast data generated by the Navy's NOGAPS capability at the Fleet Numerical Oceanography Center in Monterey, California. The NOGAPS data is made available through distribution software called NODDS which operates on an IBM-PC-compatible computer. CRC wrote and tested PC-based routines to arrange downloaded NOGAPS data into the east and north wind tables previously described. We then transferred the tables to the Macintosh using Apple File Exchange, a conversion utility which is routinely supplied with Macintosh system software. The source code for the PC-based routines has been included with the BDPS source code in the Analyst Manual appendix.

Though the wind table option was designed for use with NOGAPS data, any other data which is similarly formatted may be incorporated by using this atmosphere model option. For example, if a user had access to forecast wind data from another source or to archived wind data, the user could arrange the data into the two wind files and then use that data with BDPS. Furthermore, a user with access to a climate model could format the model's output appropriately to use that climate model data with BDPS.

2.2 CLIMATE DATA OPTION

This atmosphere model option was provided to give the BDPS user access to an empirical climate model. The model is based on NASA's Global Reference Atmosphere Model (GRAM), which is an empirical FORTRAN computer simulation of the earth's atmosphere. It was developed by the Georgia Institute of Technology under contract to NASA's Marshall Space Flight Center. Justus [2] summarizes the capabilities and operations of GRAM. The latest version of GRAM which was available during our development was the 1988 version (GRAM-88) [3].

The GRAM feature which is most significant to BDPS is that GRAM provides a worldwide, 12-month database of atmospheric properties including wind speed and wind azimuth. The wind data which GRAM provides may therefore be considered to be 4-dimensional because it is a function of latitude, longitude, altitude, and time of year. The empirical data is segmented into twelve files (one for each month) and is stored in a binary form for fastest access during execution of the model. The size of each of these files is approximately 3 megabytes.

CRC had to modify GRAM to make it suitable for use as an atmospheric model in BDPS. The baseline version of GRAM available from NASA operates in "batch" mode, whereby position input is supplied and GRAM then produces an output file containing an atmospheric profile for that position. CRC learned that Mr. Larry Schilling, working on the National AeroSpace Plane (NASP) program at NASA/Ames-Dryden, had modified GRAM-88 to work in an "interactive" mode with a NASP trajectory simulation program. Mr. Schilling was able to obtain improved performance by replacing the worldwide, empirical, monthly data files with smaller empirical data sets which only contained data for the contiguous United States (CONUS). Though GRAM-88 was designed to access the large empirical files from disk, Mr. Schilling loaded his CONUS files into memory, thereby optimizing atmosphere data access during the execution of his NASP simulation. Mr. Schilling modified the GRAM-88 structure and integrated it within his simulation to provide an interactive capability in which GRAM-88 was called at each integration step to provide new atmosphere data to be included in the equations of motion for the NASP. Because of the performance improvements and the throughput of the NASP simulation host computer, Mr. Schilling named his version "RTGRAM," which is an abbreviation for "Real-Time GRAM."

As shown in Figure 2.2-1, the climate model which CRC developed for BDPS used significant portions from both of the versions of GRAM which we obtained from NASA. From the perspective of BDPS requirements, each version had advantages and disadvantages. GRAM-88 had the advantage of worldwide data and the disadvantage of the batch mode of operation. RTGRAM had the advantage of interactive operation with a flight simulation and the disadvantage of empirical data limited to CONUS. Therefore, CRC's climate model for BDPS was built around the interactive structure of RTGRAM while using the worldwide empirical data files from GRAM-88. Though the CONUS data access techniques of RTGRAM would have provided a faster-running climate model, CRC could not reasonably expect to be able to load the entire worldwide file into memory on an average Macintosh computer. The worldwide file was required (and the CONUS file was inadequate) because DARPA personnel had advised us that, for political considerations, the balloon systems for which BDPS was intended would likely be tested in the southern hemisphere.

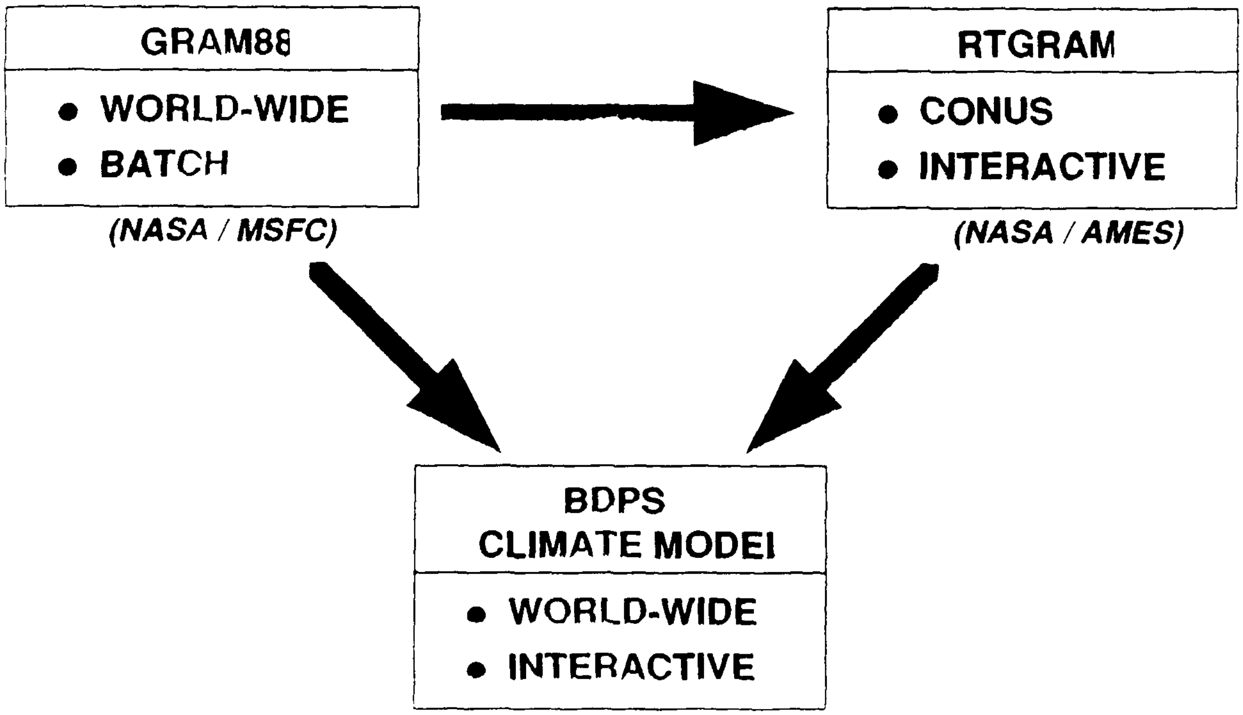


Figure 2.2-1. BDPS Climate Model Heritage

The merging of the two versions of GRAM was a difficult task because of the poor quality of the GRAM source code. The resulting source code is included in the Analyst Manual appendix. CRC had hoped to acquire GRAM-90, but its release was delayed by NASA until one month from the end of our Phase II contract. GRAM-90 offers improvements in its source code quality and in its southern hemisphere data. The features of GRAM-90 are described in publications by Justus [4] and [5].

3.0 BDPS TRAJECTORY MODEL

The second technical objective to be met was the development of a trajectory model for use in BDPS. The capability which was demonstrated in Phase I was inappropriate for Phase II for two reasons. First, the trajectory model was developed using a simulation framework which was proprietary to CRC. Since the Phase II deliverable product includes sour code, CRC had to replace the proprietary portions of the trajectory model. Second, the Phase I approach modeled balloon dynamics to a level of detail which was inappropriate for a deployment tool on a Macintosh host. The Phase II deliverable product had to be streamlined in order to provide a useable tool on a platform with much less computational throughput than the platform used for the Phase I project.

The two major design constraints in the development of the trajectory model for BDPS are that (1) the model should include the detailed effects of buoyancy variations through the balloon's ascent and the effects of winds produced by the atmosphere model and (2) the level of modeling detail should be minimized to produce a tool that is useable on the Macintosh. The following paragraphs describe the approach CRC took in balancing the high fidelity and low detail requirements.

In the Phase I literature survey, CRC found that balloon designers already had detailed thermodynamic models [6], [7] of balloon ascent (vertical motion). However, we found no published information about balloon simulations which considered horizontal motion. Given this information and DARPA's stated need for a deployment tool, CRC decided to focus on wind-induced motion as the major consideration for BDPS.

As we were formulating ideas and approaches for a BDPS trajectory model, we noticed a consistent trend in the output from a number of different test runs produced with our Phase I tool on the VAX. Figures 3.0-1 through 3.0-4 show sample wind profiles produced under four separate sets of circumstances: Wallops Island, Virginia in January; Wallops Island in August; Vandenberg AFB, California in January, and Vandenberg AFB in August. We noted that the four wind magnitude profiles showed significant variations from each other. However, Figures 3.0-5 through 3.0-8 show that the vertical motion for a single balloon configuration in each of those four circumstances was essentially constant. A comparison of the numbers in the various plots revealed a maximum difference on the order of 1%.

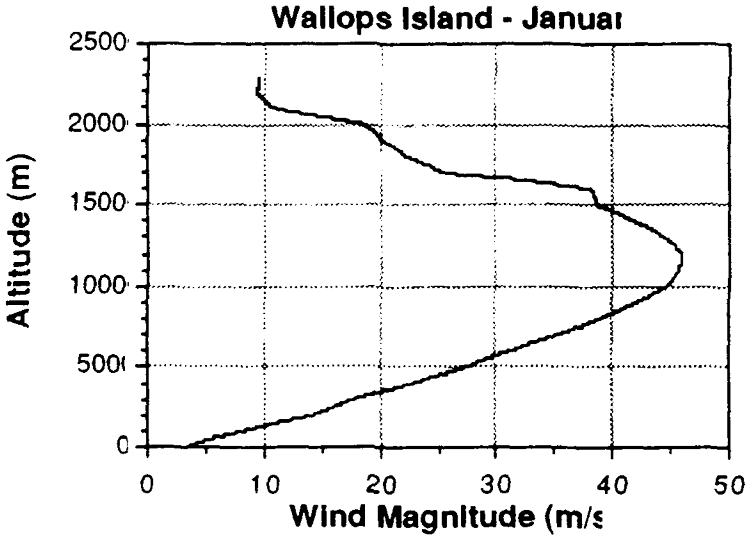


Figure 3.0-1. Wind Profile for Wallops Island in January

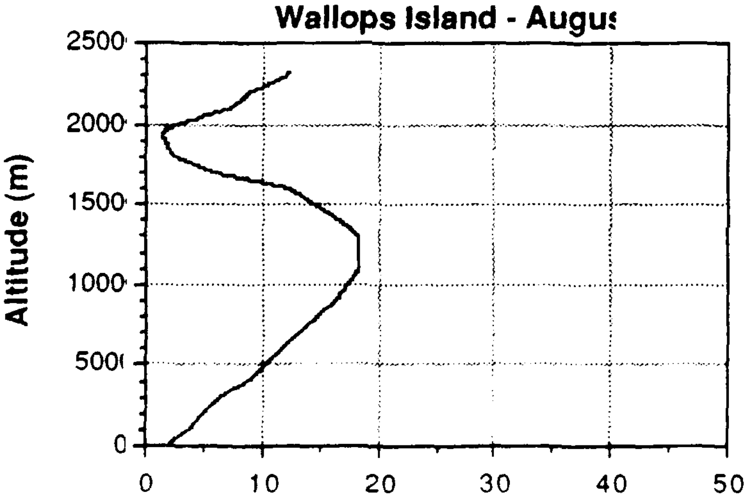


Figure 3.0-2. Wind Profile for Wallops Island in August

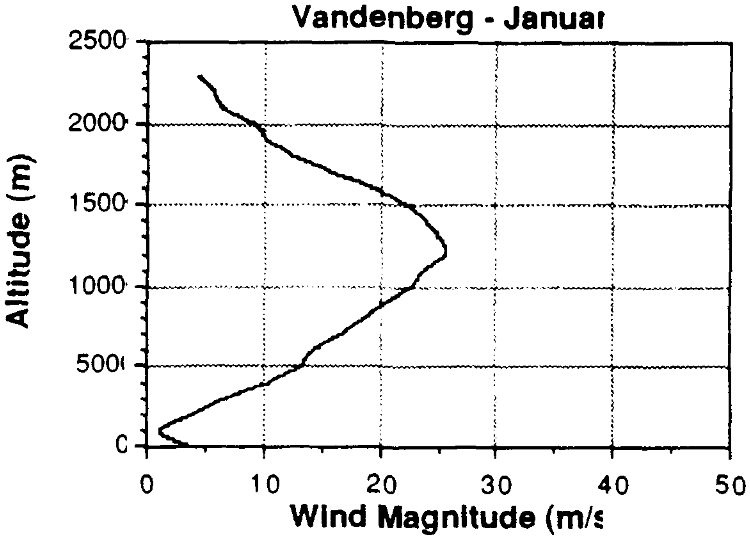


Figure 3.0-3. Wind Profile for Vandenberg AFB in January

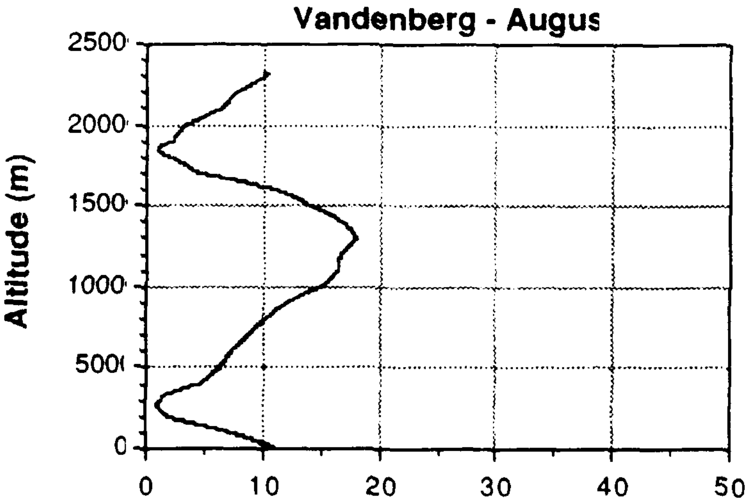


Figure 3.0-4. Wind Profile for Vandenberg AFB in August

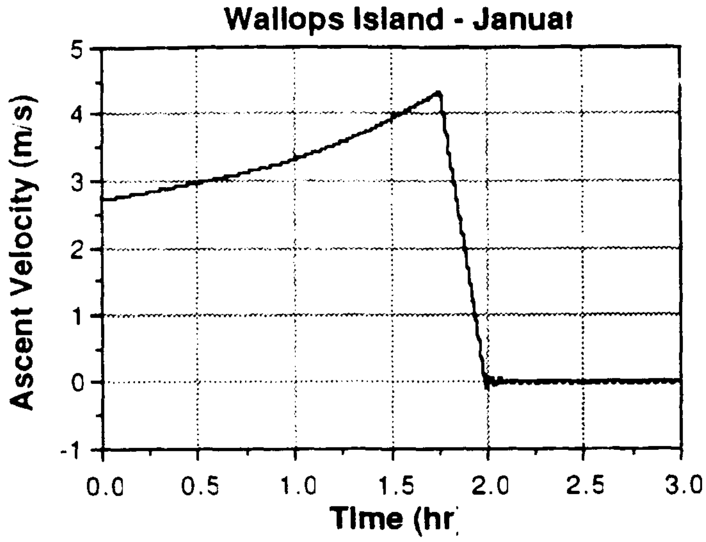


Figure 3.0-5. Ascent Profile for Wallops Island in January

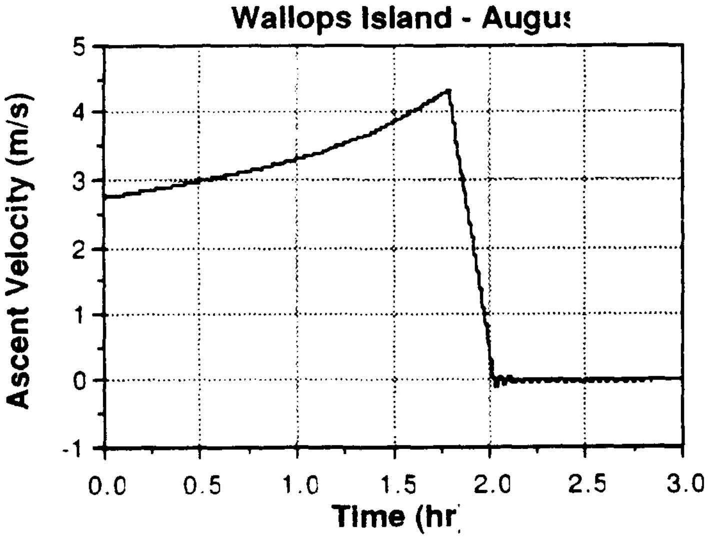


Figure 3.0-6. Ascent Profile for Wallops Island in August

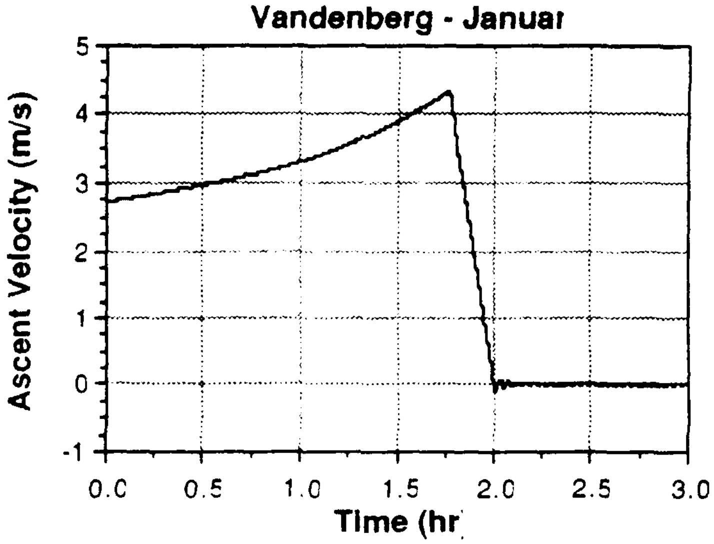


Figure 3.0-7. Ascent Profile for Vandenberg AFB in January

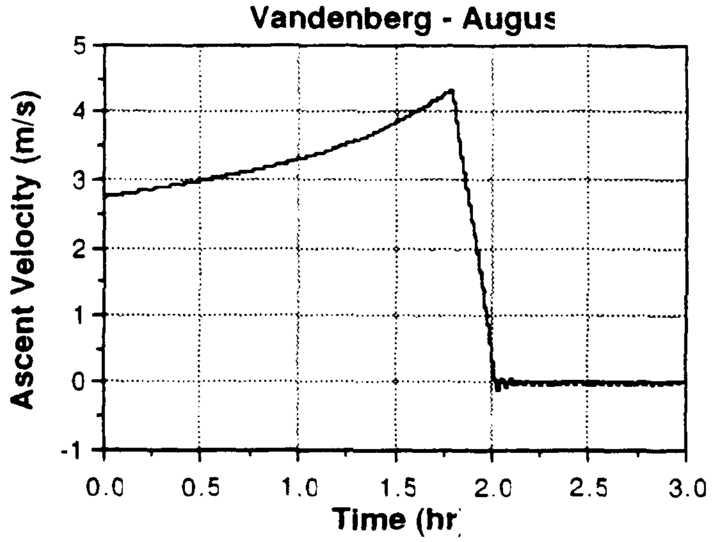


Figure 3.0-8. Ascent Profile for Vandenberg AFB in August

From the analysis of the data depicted in Figures 3.0-1 through 3.0-8, we concluded that the vertical motion of a balloon configuration is essentially independent of the horizontal wind profile with which the balloon motion is simulated. Detailed thermodynamic models [6], [7] may be used to generate the vertical motion profile for a particular balloon configuration. Furthermore, Fichtl [8] had stated that a balloon's horizontal wind-relative velocity (i.e., the difference in the balloon velocity and the wind velocity) is zero after a short time for settling oi transients. For the design of the BDPS trajectory model, we decided that the balloon's three-component velocity vector could be formed by the combination of the east and north wind velocity vectors taken from the atmosphere model (either the wind table or climate option) and the vertical velocity which may be generated offline from a more detailed, balloon design tool. By decoupling the vertical and horizontal velocity of the balloon motion, CRC has produced a simplified trajectory model which simultaneously satisfies the two design constraints stated above.

CRC tested the results of the simplified trajectory model introduced herein against the Phase I capability. The results showed acceptable agreement between the two methods. The largest observed percent difference between the two methods was less than 1%.

4.0 BDPS MACINTOSH INSTALLATION

The third technical objective to be met was the installation of the BDPS on the Macintosh. This involved four steps which are listed below and described in the following paragraphs: (1) Transfer source code, climate data files; (2) Compile and link FORTRAN code on Macintosh; (3) Duplicate sample test cases; and, (4) Add graphical user interface.

4.1 TRANSFER SOURCE AND DATA

CRC used standard file transfer tools (Kermit, Xmodem, etc.) to transfer the FORTRAN source code for BDPS from the VAX to the Macintosh. The transfer of the twelve monthly climate files, however, was more involved. For efficiency reasons already discussed, the data are stored in binary form for use by GRAM. Each of the files had to be converted to an ASCII representation on the VAX, transferred in ASCII form to the Macintosh, and then restored to binary form on the Macintosh. The conversion on the VAX and the subsequent restoration on the Macintosh were both accomplished by using very simple FORTRAN programs which were compiled and linked on the respective machines. While the binary form of each climate file occupied ~3 megabytes, the ASCII form required approximately 10 megabytes per file.

4.2 COMPILE AND LINK ON MACINTOSH

The VAX environment for BDPS development used the VMS 5.4 operating system, the VAX FORTRAN compiler, and the VAX Symbolic Debugger on a VAX 3900 machine with DEC VT-240 terminals. CRC used two Macintosh development environments for BDPS. The first was a Macintosh IIfx running System 6.0.7. On the IIfx, CRC used Apple's Macintosh Programmer's Workshop (MPW) version 3.1, the Language Systems FORTRAN compiler version 2.1, and Apple's *Symbolic Application Debugging Environment* (SADE) version 1.1. The second system was a Macintosh SE/30 running System 7.0, MPW 3.2, SADE 1.3, and the same FORTRAN compiler.

CRC encountered numerous problems in the creation of a BDPS version on the Macintosh. We had already known that the VAX and the Macintosh were internally different with respect to numeric data representation and alignment. However, the poor quality of the GRAM source code proved to be a major obstacle in producing a working BDPS version on the Macintosh. In particular, the GRAM source code flagrantly violates professional FORTRAN programming standards in its haphazard use of common blocks. The GRAM code had the distinct appearance that the various subroutines had been written by different people who engaged in little, if any, coordination while developing their separate pieces.

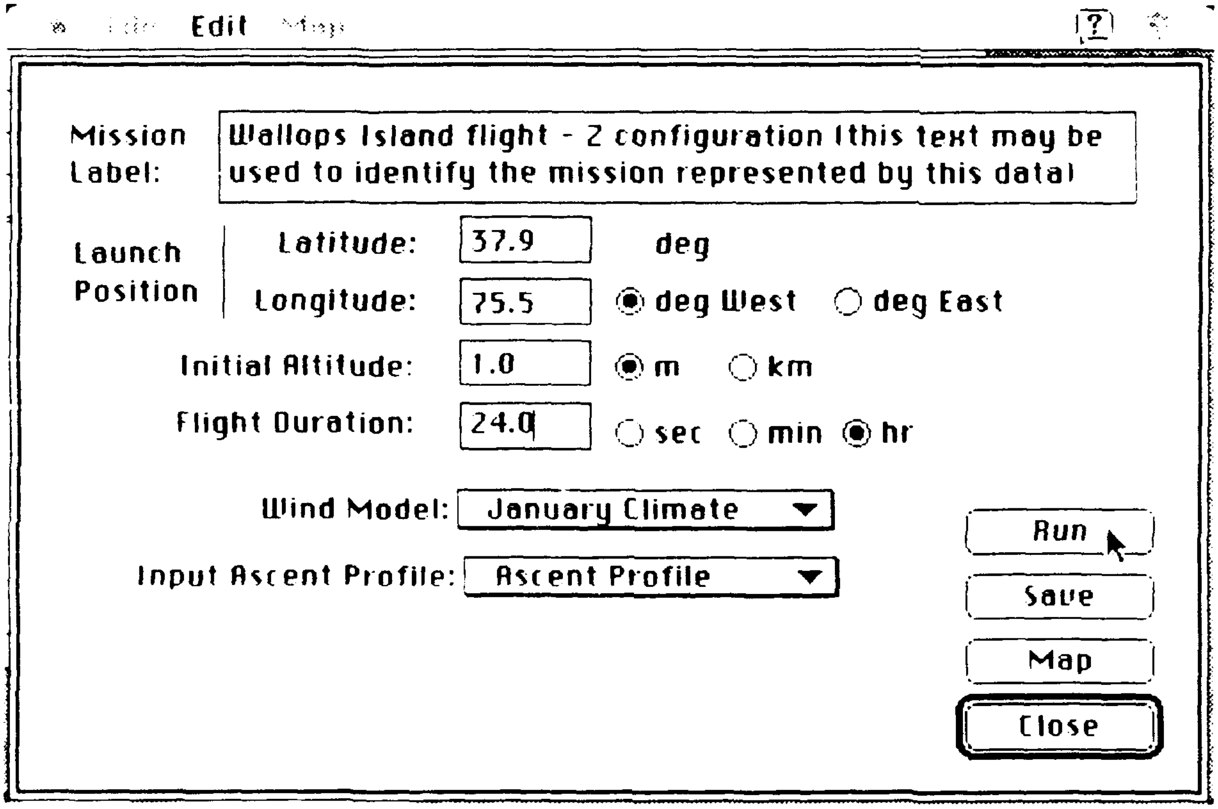
The SADE debugger was of some help in tracing problems on the Macintosh. However, our use of SADE and the VAX debugger in simultaneous debugging sessions consistently convinced us that the SADE tool, while helpful, was not as robust as the VAX debugger. The net result of this difference is that we spent more time debugging on the Macintosh than we would have otherwise. We are encouraged, however, by some substantial improvements in the functionality of SADE 1.3 as compared to that of SADE 1.1.

4.3 DUPLICATE VAX RESULTS ON MACINTOSH

Once we had a fully functional version of BDPS on the Macintosh, we conducted several tests using the same data files on each of the two platforms. The results of the tests matched each other except for discrepancies in the lower-order digits of parameters. We attribute this difference in precision to the different mechanisms which the two platforms use in floating-point calculations. The VAX uses 32-bit representations for *single precision* throughout its operations. However, the Macintosh converts the 32-bit floating-point numbers to and from 80-bit numbers for operations which involve the 68882 floating-point co-processor in the IIfx and the SE/30.

4.4 ADD GRAPHICAL USER INTERFACE

In the statement of work for the Phase II contract, CRC stated that the interface for the Macintosh-based BDPS tool would be minimal in that it would not be designed to have the appearance and operation of well-known, commercially available Macintosh applications. However, CRC acquired a copy of Prototyper 3.0, which is a Macintosh user-interface development tool. We used this tool to create a Macintosh dialog box for configuring a BDPS run. The dialog box is shown in Figure 4.4-1. The most significant feature of the tool is that it can then generate compilable source code for the dialog items and their operation. CRC used the tool to generate Pascal source code and then converted the code to FORTRAN for consistency with the majority of the BDPS code. The operation of the user interface will be described in the User Manual appendix.



Mission Label: Wallops Island flight - 2 configuration (this text may be used to identify the mission represented by this data)

Launch Position
Latitude: 37.9 deg
Longitude: 75.5 ● deg West ○ deg East

Initial Altitude: 1.0 ● m ○ km

Flight Duration: 24.0 ○ sec ○ min ● hr

Wind Model: January Climate ▼

Input Ascent Profile: Ascent Profile ▼

Figure 4.4-1. Sample Dialog Box for BDPS Run Configuration

5.0 BDPS MACINTOSH DISPLAY

The fourth technical objective to be met was the development of a capability to display the drift pattern output on the Macintosh screen. During Phase I, CRC had used a VAX-based graphics package which generated pen-plotter output. For Phase II, the goal was to provide a capability on the Macintosh that would be an integrated part of the Macintosh-based BDPS and would produce drift pattern graphics that could be imported into commercially available Macintosh graphics packages.

CRC chose to save the on-screen drift patterns in the PICT file format which is supported by the major graphics packages. With this approach, the BDPS user can predict a drift pattern, display it on the Macintosh screen, and save it in a graphics file. The user can then use a commercial graphics package to open the saved graphic, annotate it as desired, and use the commercial package's printing capability to generate a hardcopy of the drift pattern.

The drift pattern latitude-longitude format was developed in Phase I and retained for use in Phase II. The display includes a world-map background which shows the physical outlines of the world's major land masses along with several islands. A sample display is shown in Figure 5.0-1.

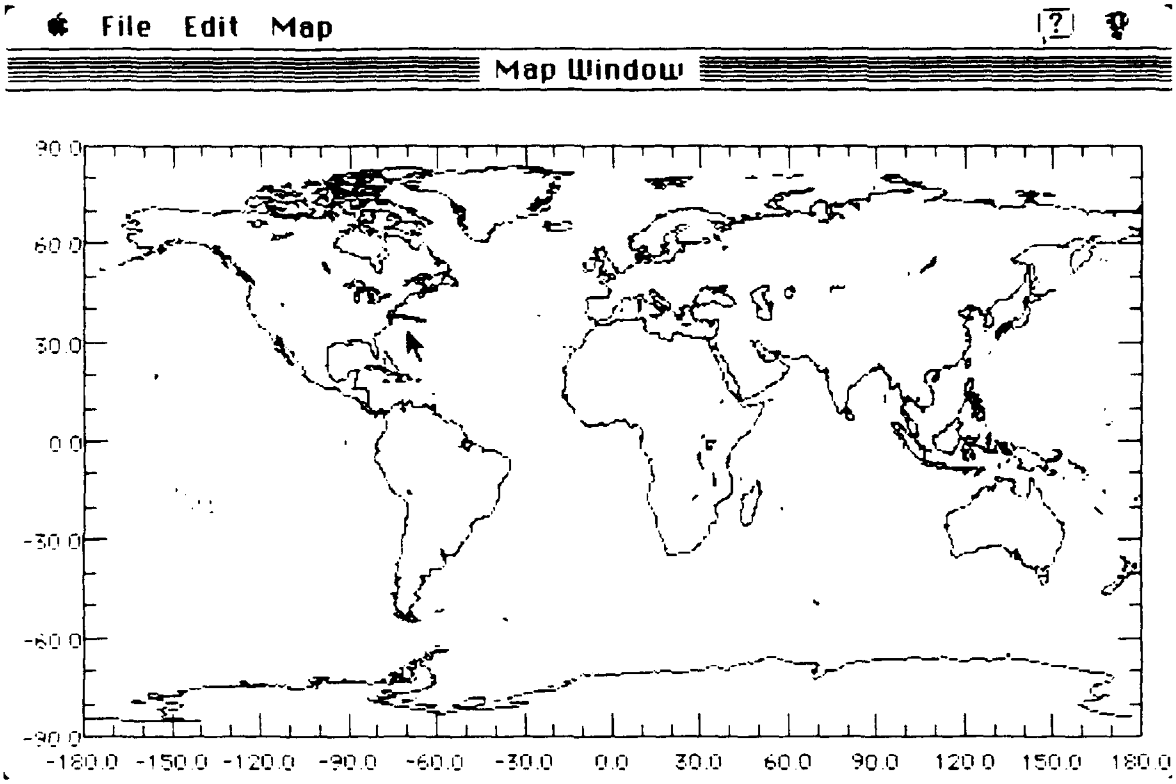


Figure 5.0-1. Sample Macintosh Drift Pattern Display

The on-screen display was developed by making FORTRAN calls to the Macintosh utility library which is present in the ROM and system software of every Macintosh. These routines are extensively documented in Inside Macintosh, volumes I through V [9]. The source code for these display routines is included in the Analyst Manual appendix.

6.0 BDPS DOCUMENTATION

The fifth and final technical objective to be met was the development of documentation for BDPS. CRC chose to present the BDPS information from two perspectives: an Analyst Manual and a User Manual. The Analyst Manual presents the engineering design approach which CRC took in developing BDPS. The Analyst Manual presents this information by listing the source code which was used to generate the BDPS application on the Macintosh. The User Manual describes the operation of the drift pattern data generation and display modes of BDPS. Both of these manuals are included as appendices to this final report.

7.0 CONCLUSIONS AND RECOMMENDATIONS

CRC has successfully developed a Balloon Drift Pattern Simulation (BDPS) tool for use on the Macintosh computer. The BDPS tool is operated through a graphical interface similar to commercial Macintosh software. The drift pattern data is generated by using one of two options for supplying wind data: a climate model based on NASA'a Global Reference Atmosphere Model, or a table-driven wind model through which the BDPS user can provide wind data from another source. The resulting balloon drift pattern may be displayed on-screen and then imported into one of several commercial Macintosh graphics programs for annotation and/or printing. All source code for the BDPS tool has been made available through an appendix to this report.

The sole recommendation which CRC has to make regarding BDPS is that the performance be improved to make the tool more convenient to use. CRC has tested BDPS on a Macintosh SE/30 and on a Macintosh IIfx. On both systems, the process of generating drift pattern data (prior to the display of the drift pattern) requires several minutes. Our testing indicates that the GRAM model's heavy dependence on external files is the primary performance bottleneck. This problem could be mitigated by either of two approaches or by the combination of both approaches. The first approach would be to restructure the GRAM model to produce more efficient runtime operation. This process should begin with GRAM-90, NASA's newest version, which was not publically released until the end of CRC's period of performance in the development of BDPS. The second approach would be to rehost BDPS on a more powerful computer such as an engineering workstation. Running BDPS on a more powerful computer would improve performance regardless of whether GRAM had been restructured.

8.0 REFERENCES

[1] Coleman Research Corporation, "Expendable Air Vehicles / High Altitude Balloon Technology Final Technical Report," 28 February 1989, CHR/89-1909.

[2] Justus, C.G., Fletcher, G.R., Gramling, F.E., and Pace, W.B., "The NASA/MSFC Global Reference Atmosphere Model - MOD 3 (with Spherical Harmonic Wind Model)," NASA CR-3256, 1980.

[3] Justus, C.G., Alyea, F.N., Cunnold, D.M., Blocker, R.S., and Johnson, D.L., "GRAM-88: Improvements in the Perturbation Simulations of the Global Reference Atmospheric Model," NASA Marshall Space Flight Center Memorandum ES44-11-9-88, November 1988.

[4] Justus, C., and Johnson, D., "Extensive Middle Atmosphere (20-120 km) Modification in the Global Reference Atmospheric Model (GRAM-90)," 28th Aerospace Sciences Meeting, AIAA 90-0477, January 1990.

[5] Justus, C.G., Alyea, F.N., Cunnold, D.M., Jeffries, W.R. III, and Johnson, D.L., "The NASA/MSFC Global Reference Atmospheric Model – 1990 Version (GRAM-90)", NASA TM 4268, April 1991.

[6] Carlson, L.A., and Horn, W.J., "A Unified Thermal and Vertical Trajectory Model for the Prediction of High Altitude Balloon Performance," Texas Engineering Experiment Station Report TAMRF-4217-81-01, June 1981.

[7] Horn, W.J., and Carlson, L.A., "THERMTRAJ: A FORTRAN Program to Compute the Trajectory and Gas and Film Temperatures of Zero Pressure Balloons," Texas Engineering Experiment Station Report TAMRF-4217-81-02, June 1981.

[8] Fichtl, G.H., "Spherical Balloon Response to Three-Dimensional Time-Dependent Flows," NASA TN D-6829, July 1972.

[9] Apple Computer, Inc., Inside Macintosh, Volumes I-V, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1985.

THIS PAGE INTENTIONALLY LEFT BLANK

10.0 BDPS ANALYST MANUAL

The purpose of this appendix to the final report is to provide insight into Coleman Research Corporation's engineering approach and software implementation for the Balloon Drift Pattern Simulation (BDPS). Because the main body of the final report presents sufficient information about the engineering approach, this appendix presents the BDPS software details through the inclusion of the FORTRAN source code and associated utilities.

10.1 BDPS GENERATION SCRIPTS

As described in section 4.2, CRC used Apple's Macintosh Programmer's Workshop (MPW) and the Language Systems Corporation FORTRAN compiler. CRC automated the compilation and linking processes with scripts which are included h re.

10.1.1 BDPS Compilation Script

CRC used the following compilation script to in the generation of the BDPS object code modules. The script uses three directories: a source-code directory (in which FORTRAN filenames end with ".f", an include-file directory in which the filenames end with ".inc", and an object-code directory in which the compiler's output files are placed. Two GRAM-related files, "run.f" and "gen4d.f," are compiled with default 4-byte integer sizes. The other files are compiled with default 2-byte integer sizes because that size is more commonly used with Macintosh Toolbox arguments.

```
#
# bdpsCompile -   compile any source code targeted for bdps if either the
#                 object code doesn't exist or the source code is newer
#
#   Targets every file whose name ends in .f
#   in the source code directory
#

SetDirectory Mars:darpa:MacBDPS:SourceFiles:

set FortranOptions          "-b -i Mars:darpa:MacBDPS:IncludeFiles: ∂
                             -i2 -mc68030 -mc68882 -nodyn -opt=1    ∂
                             -saveall -sane -sym -u"
set FortranOptionsAlternate "-b -i Mars:darpa:MacBDPS:IncludeFiles: ∂
                                 -mc68030 -mc68882 -nodyn -opt=1    ∂
                             -saveall -sane -sym -u"

# compile run.f and gen4d.f with alternate compilation (for I*4)

if `exists Mars:darpa:MacBDPS:ObjectFiles:run.f.o`
   if  Newer run.f Mars:darpa:MacBDPS:ObjectFiles:run.f.o
      fortran run.f {FortranOptionsAlternate} ∂
                   -o Mars:darpa:MacBDPS:ObjectFiles:run.f.o
   end if
else if not `exists Mars:darpa:MacBDPS:ObjectFiles:run.f.o`
      fortran run.f {FortranOptionsAlternate} ∂
```

```
                    -o Mars:darpa:MacBDPS:ObjectFiles:run.f.o
end if

if `exists Mars:darpa:MacBDPS:ObjectFiles:gen4d.f.o`
   if `Newer gen4d.f Mars:darpa:MacBDPS:ObjectFiles:gen4d.f.o`
      fortran gen4d.f {FortranOptionsAlternate} ∂
                      -o Mars:darpa:MacBDPS:ObjectFiles:gen4d.f.o
   end if
else if not `exists Mars:darpa:MacBDPS:ObjectFiles:gen4d.f.o`
      fortran gen4d.f {FortranOptionsAlternate} ∂
                      -o Mars:darpa:MacBDPS:ObjectFiles:gen4d.f.o
end if

# compile everything else (run.f and gen4d.f will be checked and passed)

for file in ≈.f
    set ObjectFile "Mars:darpa:MacBDPS:ObjectFiles:{file}.o"
    set SourceFile "Mars:darpa:MacBDPS:SourceFiles:{file}"
    if `exists {ObjectFile}`
        if `Newer {SourceFile} {ObjectFile}`
            fortran {SourceFile} {FortranOptions} -o {ObjectFile}
        end if
    else if not `exists {ObjectFile}`
         fortran {SourceFile} {FortranOptions} -o {ObjectFile}
    end if
end
```

10.1.2 BDPS Linking Script

CRC used the following linking script to accomplish two objectives. First, the script compiles the BDPS resource file (if necessary). Then, the script links the previously compiled object code files with the required libraries to produce an executable application.

```
# bdpsLink - link the object code in Mars:darpa:MacBDPS:ObjectFiles: to form
#               application bdps
#

set RawRezFile      "Mars:darpa:MacBDPS:RezFiles:bdps.r"
set CompiledRezFile "Mars:darpa:MacBDPS:bdps.rsrc"

if `Newer {RawRezFile} {CompiledRezFile}`
  SetDirectory Mars:darpa:MacBDPS:RezFiles
  rez -o Mars:darpa:MacBDPS:bdps.rsrc -c MDoF bdps.r ∂
      -i Mars:MPW:Interfaces:RIncludes:
end if

SetDirectory Mars:darpa:MacBDPS:

duplicate -y bdps.rsrc bdps

Link -ac 4 -ad 4 -f -srt -sym on -w -o bdps         ∂
     Mars:darpa:MacBDPS:ObjectFiles:≈.f.o           ∂
     {Libraries}Runtime.o                           ∂
     {Libraries}Interface.o                         ∂
     {FLibraries}OutPWStubs.o                       ∂
     {FLibraries}FortranLib.o                       ∂
     {FLibraries}IntrinsicLibFPU.o                  ∂
     {FLibraries}FSANELibFPU.o
```

10.2 BDPS INCLUDE FILES

This section lists the contents of the “include” files which were made available to the FORTRAN compiler during compilation of the BDPS FORTRAN source code. Include files were generally used for the definition of common blocks and for the declaration of variable and static parameters which were referenced within multiple source code files. The filename for each include file is given in the commented first line of each file.

```
c.....Alert.inc

c.....Alert declarations

      integer*2          rAboutAlert

c.....Alert pre-settings

      parameter          ( rAboutAlert        = 128 )
```

```
c.....AppleMenu.inc

c.....Apple menu declarations

      integer*2              nAppleItems
      integer*2              AppleItemAboutBDPS
      integer*2              AppleItemHelp
      integer*2              AppleMenuID
      record / MenuHandle /  AppleMenuHndl

c.....Apple menu common block

      common / AppleMenu /   AppleItemAboutBDPS,      AppleItemHelp,
     .                       AppleMenuID,
     .                       AppleMenuHndl

c.....Apple menu pre-settings

      parameter              ( nAppleItems          =   2 )
      parameter              ( AppleItemAboutBDPS   =   1 )
      parameter              ( AppleItemHelp        =   2 )
      parameter              ( AppleMenuID          = 128 )
```

```
c.....CrvDat.inc

c.....Map curve parameters

      parameter          (ncrvmx=4)

      common / crvdat /  ipvari,     ipvard,     ipvarg,     idrlin,
     .                   lintyp,     idrsym,     symtyp,     iplogx,
     .                   iplogy,     ipstep,     DshMsk,     PixCnt,
     .                   ighoff,
     .                   gradat

      integer*2          ipvari(ncrvmx)
      integer*2          ipvard(ncrvmx)
      integer*2          ipvarg(ncrvmx)
      integer*2          idrlin(ncrvmx)
      integer*2          lintyp(ncrvmx)
      integer*2          idrsym(ncrvmx)
      integer*2          symtyp(ncrvmx)
      integer*2          iplogx(ncrvmx)
      integer*2          iplogy(ncrvmx)
      integer*2          ipstep(ncrvmx)
      integer*2          DshMsk(ncrvmx)
      integer*2          PixCnt(ncrvmx)
      integer*2          ighoff(ncrvmx)

      real*4             gradat(ncrvmx)
```

```
c.....DefLim.inc

c.....Map data common block

      common / DefLim / LatMin,     LatMax,     LatDivMj,   LatDivMi,
     .                  LngMin,     LngMax,     LngDivMj,   LngDivMi

      real*4             LatMin
      real*4             LatMax
      real*4             LatDivMj
      real*4             LatDivMi
      real*4             LngMin
      real*4             LngMax
      real*4             LngDivMj
      real*4             LngDivMi
```

```
c.....EditMenu.inc

c.....Edit menu declarations

      integer*2                nEditItems
      integer*2                EditItemUndo
      integer*2                EditItemCut
      integer*2                EditItemCopy
      integer*2                EditItemPaste
      integer*2                EditItemClear
      integer*2                EditMenuID
      record / MenuHandle /    EditMenuHndl

c.....Edit menu common block

      common / EditMenu /   EditItemUndo,         EditItemCut,
     .                      EditItemCopy,         EditItemPaste,
     .                      EditItemClear,
     .                      EditMenuID,
     .                      EditMenuHndl

c.....Edit menu pre-settings

      parameter            ( nEditItems        =    5 )
      parameter            ( EditItemUndo      =    1 )
      parameter            ( EditItemCut       =    2 )
      parameter            ( EditItemCopy      =    3 )
      parameter            ( EditItemPaste     =    4 )
      parameter            ( EditItemClear     =    5 )
      parameter            ( EditMenuID        =  130 )
```

```
c.....FileInfo.inc

c.....File information declarations

        logical*1               iGotOldFile
        integer* 2              RefNum

c.....File information common block

        common / FileInfo /
       &                        iGotOldFile, RefNum
```

```
c.....FileMenu.inc

c.....File menu declarations

        integer*2               nFileItems
        integer*2               FileItemNewMission
        integer*2               FileItemOpenMission
        integer*2               FileItemClose
        integer*2               FileItemSave
        integer*2               FileItemSaveAs
        integer*2               FileItemPageSetup
        integer*2               FileItemPrint
        integer*2               FileItemQuit
        integer*2               FileMenuID
        record / MenuHandle /   FileMenuHndl

c.....File menu common block

        common / FileMenu /     FileItemNewMission,     FileItemOpenMission,
       &                        FileItemClose,          FileItemSave,
       &                        FileItemSaveAs,         FileItemPageSetup,
       &                        FileItemPrint,          FileItemQuit,
       &                        FileMenuID,
       &                        FileMenuHndl

c.....File menu pre-settings

        parameter              ( nFileItems          =   8 )
        parameter              ( FileItemNewMission  =   1 )
        parameter              ( FileItemOpenMission =   2 )
        parameter              ( FileItemClose       =   4 )
        parameter              ( FileItemSave        =   5 )
        parameter              ( FileItemSaveAs      =   6 )
        parameter              ( FileItemPageSetup   =   9 )
        parameter              ( FileItemPrint       =  10 )
        parameter              ( FileItemQuit        =  12 )
        parameter              ( FileMenuID          = 129 )
```

```
c.....FntCom.inc

c.....font characteristics

        common   FntCom                     FontData,     FntNam,              FntNum

        record / FontInfo /     FontData
        string*255              FntNam
        integer*2               FntNum
```

```
c.....Globals.inc

c.....Globals declarations

      integer*4              inFront
      integer*2              SleepValue

      logical*1              gInBackground
      logical*1              gHasWaitNextEvent
      logical*1              doneFlag

      record / TEHandle   / theInput
      record / SysEnvRec  / gMac

c.....Globals common block

      common / Globals    /
     .                       SleepValue,
     .                       gInBackground,
     .                       gHasWaitNextEvent,
     .                       doneFlag,
     .                       theInput,
     .                       gMac
c.....Globals pre-settings

      parameter           ( inFront             =  -1 )
      parameter           ( SleepValue          =  40 )
```

```
c.....LatCom.inc

c     common block containing array of latitude values

      common / LatCom /                      Latitude

      real                         Latitude(13120)
```

```
c.....LngCom.inc

c     common block containing array of longitude values

      common / LngCom /                      Longitude

      real                         Longitude(13120)
```

```
c.....MapCom.inc

c.....Map data common block

      parameter           (nptsmx=2048)
      parameter           (nvrsmx=  64)

      common   Mapcom    MapWidth,   MapHeight,  MapHres,    MapVres,
     .                   DefWidth,   DefHeight,
     .                   TimeTics,   GridLines,  LimitType,
     .                   npts,       nvrs,       nMaps
c     .                   ,header

      real*4              MapWidth
      real*4              MapHeight
      real*4              MapHRes
      real*4             MapVRes
      real*4             DefWidth
      real*4             DefHeight

      integer*2          TimeTics
      integer*2          GridLines
      integer*2          LimitType

      integer*4          npts
      integer*4          nvrs
      integer*4          nMaps

c      string*255         header
```

```
c.....MapLim.inc

c.....map limits and divisions

      common / MapLim / xMapMn,     xMapMx,     xDivMj,     xDivMi,
     .                  yMapMn,     yMapMx,     yDivMj,     yDivMi,
     .                  tMapMn,     tMapMx,     tDivMj

      real*4            xMapMn
      real*4            xMapMx
      real*4            xDivMj
      real*4            xDivMi
      real*4            yMapMn
      real*4            yMapMx
      real*4            yDivMj
      real*4            yDivMi
      real*4            tMapMn
      real*4            tMapMx
      real*4            tDivMj
```

```
c.....MapMenu.inc

c.....Map menu declarations

      integer*2             nMapItems
      integer*2             itemGetNewDataSet
      integer*2             itemResizeTheMap
      integer*2             itemNewMap
      integer*2             itemSaveMap
      integer*2             itemRedraw
      integer*2             itemDone
      integer*2             MapMenuID
      logical*2             EnableTheItem
      logical*2             DisableTheItem
      record / MenuHandle / MapMenuHndl

c.....Map menu common block

      common   MapMenu      itemGetNewDataSet,      itemResizeTheMap,
     &                      itemNewMap,             itemSaveMap,
     &                      itemRedraw,             itemDone,
     &                      MapMenuID,
     &                      EnableTheItem,          DisableTheItem,
     &                      MapMenuHndl

c.....Map menu pre-settings

```

```
        parameter           ( nMapItems          =   6       )
        parameter           ( itemGetNewDataSet  =   1       )
        parameter           ( itemResizeTheMap   =   2       )
        parameter           ( itemNewMap         =   3       )
        parameter           ( itemSaveMap        =   4       )
        parameter           ( itemRedraw         =   5       )
        parameter           ( itemDone           =   6       )
        parameter           ( MapMenuID          = 131       )
        parameter           ( EnableTheItem      = .true.    )
        parameter           ( DisableTheItem     = .false.   )
```

```
c.....MBar.inc

c.....MBar declarations

        integer*2           MenuBarID
        record / MenuHandle / MenuBar

c.....MBar common block

        common / MBar       /  MenuBarID,  MenuBar

c.....MBar pre-settings

        parameter           ( MenuBarID            = 128 )
```

```
C.... NASPCOM.inc:  UNIVERSAL COMMON BLOCK LIST FOR NASP SIM.  07/28/88
C
        COMMON /ACOUT1/ CL0_2 ,CD0_2 ,CM0_2 ,CMDE_2,CMQ_2
        LOGICAL         PMAN  ,PAUTO ,AUTOP ,RMAN  ,RAUTO ,AUTOR ,
     .                  SBMAN ,SBAUTO,AUTOSB,HISPED,LOSPED
        COMMON /ACSDAT/ PMAN  ,PAUTO ,AUTOP ,RMAN  ,RAUTO ,AUTOR ,
     .                  SBMAN ,SBAUTO,AUTOSB,HISPED,LOSPED,VDOTC ,
     .                  ALPNOM,ANZC  ,PHIC  ,PHICG ,VREFG ,ACSQC ,
     .                  ACSPC ,ACSSBC
        COMMON /ACSGAN/ AKVDOT,AKALP ,AK    ,AKNZ  ,AKPHI ,AKSB  ,
     .                  AKISB ,SBLIMI,SBIN  ,SBOUT ,SBREF
        COMMON /ACSOUT/ A01JN1,A02OT1,A03JN1,A04JN1,A05OT1,A06OT1,
     .                  A07OT1,A08OT1,A09JN1,A10JN1,A11OT1,A31OT1,
     .                  A32JN1,A33OT1,A34OT1,A61JN1,A62OT1,A63OT1,
     .                  A64OT1,A65JN1,A66OT1,A67OT1
        COMMON /ACTCON/ DELA  ,DELRP ,DELRN ,DELPP ,DELPN ,DELHY ,
     .                  DERA  ,DERRP ,DERRN ,DERPP ,DERPN ,DERHY ,
     .                  DR1A  ,DR1RP ,DR1RN ,DR1PP ,DR1PN ,DR1HY ,
     .                  DR2A  ,DR2RP ,DR2RN ,DR2PP ,DR2PN ,DR2HY ,
     .                  DRILA ,DRILRP,DRILRN,DRILPP,DRILPN,DRILHY,
     .                  DRIRA ,DRIRRP,DRIRRN,DRIRPP,DRIRPN,DRIRHY,
     .                  DROLA ,DROLRP,DROLRN,DROLPP,DROLPN,DROLHY,
     .                  DRORA ,DRORRP,DRORRN,DRORPP,DRORPN,DRORHY
        COMMON /ACTDAT/ DELDAT(10),DERDAT(10),DR1DAT(10),DR2DAT(10),
     .                  DRILDT(10),DRIRDT(10),DROLDT(10),DRORDT(10)
        COMMON /AFILTK/ A64F1K
        COMMON /ALGAIN/ GKHDOT,GHLIM ,GKH   ,GKL   ,GAL   ,GLLIM ,
     .                  GKYDOT,GYMAX ,GKY   ,GPHIMX,GKHEPP
        COMMON /ALGANA/ GMCHBP(2),GKYDTA(2),GKYA(2),GYMAXA(2)
        COMMON /ALGDAT/ DHDNOM,GAMMA2,GAMMA4,GDX   ,GHF   ,GP    ,
     .                  GVIC  ,GXK   ,HAL   ,HDBIAS,HDECAY,HDEPP ,
     .                  HDOTF ,HDOTIC,HDREF ,HERR  ,HERRX ,HFF   ,
     .                  HFINAL,HFLARE,HMING ,HREF  ,HREF4 ,HTD   ,
     .                  HWHEEL,IPHASE,RANGE ,SIGMA ,TAU4  ,VGRND ,
     .                  XAIM2 ,XAIM4 ,XEXP  ,YDOTRW,HDTDCA
```

```
          COMMON /ALTFUN/ A      ,RHO    ,G      ,PA     ,TEMPR
          COMMON /ARFMIC/ AMSSIC,FMSSIC,FMSMAX,AIXIC ,AIYIC ,AIZIC ,AIXZIC
          LOGICAL         IFURUN
    C##      INTEGER*8       IACNT ,IBCNT
          COMMON /AROFLG/ IFURUN,IACNT ,IBCNT ,INTCNB,ITAERO
          COMMON /AROUT2/ CLLB  ,CLLDA ,CLLDR ,CLLP  ,CLLR  ,
         .                CLNB  ,CLNDA ,CLNDR ,CLNP  ,CLNR  ,
         .                CYB   ,CYDA  ,CYDR
          COMMON /ATOVAR/ ITYPE,GS
          COMMON /A3OUT1/ CL0_3 ,CD0_3 ,CDI_3 ,CM0_3
          COMMON /BLOUT1/ CL0_B ,CD0_B ,CM0_B ,CMDE_B,CMQ_B
          COMMON /CGSHFT/ DELX  ,DELY  ,DELZ
          CHARACTER*4     DNAMES     ,DNAME ,CMANDS    ,CMAND
          COMMON /CHNGDS/ DNAMES(100),DNAME ,CMANDS(20),CMAND ,IDSPGE
          COMMON /CICDAT/ ICN   ,RATIC
          COMMON /CLCOUT/ CLL   ,CM    ,CLN   ,CD    ,CL    ,CY    ,
         .                CN    ,CA    ,XLCD
          COMMON /CONCOM/ DAC   ,DEC   ,DRC   ,DELC  ,DERC  ,DR1C  ,
         .                DR2C  ,DRILC ,DRIRC ,DROLC ,DRORC ,THRC
          COMMON /CONDAT/ RI1   ,RI2   ,RI3   ,RI4   ,RI5   ,RI6   ,
         .                SB    ,SC    ,SM
          COMMON /CONFIG/ BODLEN,CGREF ,PMAC  ,ICONFG
          LOGICAL         DIRECT
          COMMON /CONPOS/ DAP   ,DEP   ,DRP   ,DSBP  ,DBFP  ,DLGP  ,
         .                THRP  ,DIRECT
          COMMON /CONTRL/ DA    ,DE    ,DR    ,DEL   ,DER   ,DR1   ,
         .                DR2   ,DRIL  ,DRIR  ,DROL  ,DROR  ,DRI   ,
         .                DRO   ,DRV   ,DSB
          COMMON /CSINIT/ P04A  ,P06A  ,P13A  ,P13B  ,Y05OMG,Y05ZET,
         .                Y08A  ,Y10A  ,Y12A
          CHARACTER       NAMV*12,NAMI*12,ITYPVR*1
          LOGICAL*1       DEGMOD,WGHTMD,NFOUND
          COMMON /DACLIS/ VALU(16),IADR(16),NAMV(16),NAMI,DEGMOD(16),
         .                WGHTMD(16),ITYPVR(16),NFOUND,XMXVAL(16),
         .                XMNVAL(16),SCAL(16),BIAS(16),NBYTES(16)
          COMMON /DATAIN/ S     ,B     ,CBAR  ,AMSS  ,AIX   ,AIY   ,
         .                AIZ   ,AIXZ  ,AIXE
          COMMON /DBANDS/ PDBAND,RDBAND,YDBAND,DEDB1 ,DEDB2 ,DADB1 ,DADB2 ,
         .                DRDB1 ,DRDB2
          COMMON /DIRGAN/ PDIRGN,RDIRGN,YDIRGN
          LOGICAL*1       SSW,OSW
          COMMON /DISDAT/ SSW(256),OSW(256)
          REAL*8          T     ,P     ,Q     ,R     ,RUH   ,PSIH  ,
         .                WH    ,THA   ,PSI   ,PHI   ,DELR  ,LAT_RAD ,
         .                LON_RAD
    !!!      COMMON /DRVOUT/ T     ,P     ,Q     ,R     ,RUH   ,PSIH  ,
    !!!     .                WH    ,THA   ,PSI   ,PHI   ,DELR  ,LAT_RAD ,
    !!!     .                LON_RAD ,TDOT  ,PDOT  ,QDOT  ,RDOT  ,RUHDOT,
    !!!     .                PSIHDT,WHDOT ,THADOT,PSIDOT,PHIDOT,DELRDT,
    !!!     .                XLATDT,XLNGDT
    !!!      COMMON /DRVOT2/ ALP   ,ALPDOT,BTA   ,BTADOT,H     ,HDOT  ,
    !!!     .                V     ,VDOT  ,X     ,XDOT  ,Y     ,YDOT  ,VI
          REAL*8          VAL
          CHARACTER*80    LINE     ,ASCIIN,VARLEQ,VARREQ
          COMMON /DSPLAY/ LINE(23),ASCIIN,VARLEQ,VARREQ,VAL    ,IVAL
         .                IPAGE   ,INDXVP,LFTP
          COMMON /DTAHLD/ DTA(128),ATD(128),IDA(128),IAD(128)
         .                IDSTAT,IASTAT,NUMDAT,NUMADT
          COMMON /ENGCOM/ THRST ,TFACTR
          COMMON /ENGCO3/ CTA   ,CTN   ,THRSTN
          LOGICAL         ATOPHI
          COMMON /ENGDT1/ PHICMD,AC    ,AINF  ,FS    ,EISP  ,SENG  ,
         .                XEOUT ,EODRAG,C1COEF,C2COEF,C3COEF,C4COEF,
         .                ATOPHI,CAPR  ,EFFISP
```

```
      COMMON /ENGDT2/ DCMMAX,DCMFCT,DCMEO ,UWEOUT,XNOUT ,CPRS  ,PC    ,
     .                DVHCL ,BTAP  ,BTAU  ,BTASD ,BFACTR,TLIMCB,SINDVH,
     .                COSDVH
      LOGICAL         FOUT  ,NOWC
      COMMON /ENGINF/ FQNTY ,FFLOW ,FOUT  ,NOWC
      REAL*8          RLOCAL
      COMMON /EOMDAT/ RLOCAL,FXB   ,FYB   ,FZB   ,ULV   ,VLV   ,
     .                WLV   ,ULVRA ,VLVRA ,WLVRA ,AXLV  ,AYLV  ,
     .                AZLV  ,AXH   ,AYH   ,AZH   ,XL1   ,XL2   ,
     .                XL3   ,XM1   ,XM2   ,XM3   ,XN1   ,XN2   ,
     .                XN3   ,SINPSH,COSPSH
      COMMON /EPDATA/ T4T0A (2),T4T0R    ,T4T0     ,T4       ,
     .                T5T4A (2),T5T4R    ,T5T4     ,T5       ,
     .                P4P0A (2),P4P0R    ,P4P0     ,P4       ,
     .                P5P4      ,P5      ,
     .                TTEMPR    ,CONRAT  ,WDLAIR   ,
     .                WSWPA (2),WSWPR    ,WDSWDP   ,
     .                PFPT      ,PTPA    ,PTLSS    ,
     .                PCHMBR    ,
     .                XMACH (5),PAPSI
      COMMON /EPDATH/ COEF1H(2),COEF2H(4),COEF3H(2),COEF4H(4),COEF5H(4),
     .                COEF6H(3),COEF7H(7)
      COMMON /EPDATL/ COEF1L(2),COEF2L(2),COEF3L(2),COEF4L(2),COEF5L(2),
     .                COEF6L(2),COEF7L(2),COEF8L(2),COEF9L(3),COE10L(3),
     .                COE11L(3),COE12L(3),COE13L(4),COE14L(2),COE15L(2),
     .                COE16L(5)
      LOGICAL         ESSWTH
      COMMON /ESCOMN/ ESSWTH
      LOGICAL         INITLZ,HOLDIC
      COMMON /FRSTIC/ INITLZ,HOLDIC
      REAL*8          RIC1  ,RIC2  ,RIC3  ,RTG1  ,RTG2  ,RTG3  ,
     .                RNGIC ,RNGRW ,CSLTIC,SNLTIC,CSLNIC,SNLNIC,
     .                CSLTRW,SNLTRW,CSLNRW,SNLNRW
      COMMON /GCCALC/ RIC1  ,RIC2  ,RIC3  ,RTG1  ,RTG2  ,RTG3  ,
     .                RNGIC ,RNGRW ,CSLTIC,SNLTIC,CSLNIC,SNLNIC,
     .                CSLTRW,SNLTRW,CSLNRW,SNLNRW,BEAR  ,DELAZ
      COMMON /GCSOUT/ G01OT1,G02OT1,G03OT1,G04JN1,G05OT1,G06JN1,
     .                G07OT1,G08OT1,G09OT1,G10OT1,G11JN1,G12OT1,
     .                G31OT1,G32OT1,G33OT1,G34JN1,G35OT1
      LOGICAL         GEAR
      COMMON /GEARCM/ GEAR
      COMMON /GEARFM/ ALGR  ,AMGR  ,ANGR  ,FXGR  ,FYGR  ,FZGR
      COMMON /GEARIN/ XG0(3),YG0(3),ZG0(3),XKRG(3),XKD(3),XKBRK,
     .                XMUG,XMUR,XMUS(3),AMAXB
      LOGICAL         GC
      COMMON /GEAROT/ ZG(3),DZG(3),DZGDOT(3),FBG(3),FDG(3),
     .                FRG(3),FSG(3),FXBG(3),FYBG(3),GC(3),
     .                XGB(3),YGB(3),ZGB(3),PSIW(3)
      COMMON /GEREYE/ XGEAR ,ZGEAR ,XEYE  ,ZEYE
      COMMON /GFILTK/ G09F1K,G12F1K
      LOGICAL            LOGD  ,LOGU
      COMMON /GLOBAL04/  ILIST(13,62,3),IVT(3),IDEV(3),IPRI(3),
     .                   IOCD(3),IOLIST(4,5,3),ILAST(3),
     .                   ITILOC(3),I1553(576),DLK(128),RADAR(16),
     .                   IBD(32),LOGD(32),ULK(16),SCV(16),SCB(16),
     .                   SCS(16),IBU(32),LOGU(32),PMDU(256)
      COMMON /GRANDS/ U
      COMMON /GSTINP/ SIGU  ,SIGV  ,SIGW  ,DLU   ,DLV   ,DLW   ,VREF
      COMMON /GSTINT/ CU    ,CV    ,CW    ,CP    ,CQ    ,CR    ,
     .                EXU1  ,EXV1  ,EXW1  ,EXP1  ,EXQ1  ,EXR1  ,
     .                EXV2  ,EXW2  ,EXV3  ,EXW3  ,UG1   ,VG1   ,
     .                WG1   ,PG1   ,QG1   ,RG1   ,VG2   ,WG2   ,
     .                GU1   ,GV1   ,GW1   ,GP1   ,GQ1   ,GR1   ,
     .                GV2   ,GW2   ,SFV
```

```
      COMMON /GSTOUT/ ALPG  ,BTAG  ,PG    ,QG    ,RG    ,UG    ,
     .                VG    ,WG
      LOGICAL         GSTSET,GST
      COMMON /GSTUSE/ GSTSET,GST
      LOGICAL         GINIT
      COMMON /GUIDI2/ GDK1  ,GDK2  ,GDK3  ,GDK4  ,GDK5  ,GDK6  ,
     .                GDK7  ,GDK8  ,GDK9  ,GDK10 ,GDK11 ,GDK12 ,
     .                GDK13 ,GDK14 ,GDK15 ,GDK16 ,GDK17 ,XKROLL,
     .                VQ    ,XLODM ,VS    ,GINIT
      COMMON /GUIDI3/ RNGBKP(15),AMCHDA(15),QBRNML,QBRNMH
      COMMON /GUIDNC/ IGUIDE
      COMMON /GUIDO2/ THAACD,PHICD ,DD    ,THAAC1,PHICD1,DREFP ,
     .                ALDREF,HDTREF,DRAGA
      COMMON /GUIDO3/ QBRCMD,QBRNOM,QBRERR
      CHARACTER*72    HEADER
      COMMON /HDRDAT/ HEADER
      COMMON /HEATIC/ QTSIC (2),QSIC  (2),TWSIC (2),QTFIC(20),
     .                QFIC  (20),TWFIC(20)
      COMMON /HETDAT/ QTSTAG(2),QSTAG (2),TWSTAG(2),QTFLAT(20),
     .                QFLAT(20),TWFLAT(20)
      COMMON /HETDOT/ QTSDOT(2),QSDOT (2),TWSDOT(2),QTFDOT(20),
     .                QFDOT(20),TWFDOT(20)
      COMMON /HETDT1/ XMUINF,RHOLB ,ENTHI
      COMMON /HETDT2/ ENTHRS(2),ENTHST(2),ENTHWS(2),HTRANS(2),
     .                PAST(2)  ,PAW(2)   ,RHOW(2)  ,RSTMST(2),
     .                RWMW(2)  ,TMPST(2) ,TRSTAG(2),VGRAD(2) ,
     .                XMUW(2)
      COMMON /HETDT3/ SWEEP(2) ,COSSWP(2),SINSWP(2),RCURVE(2),
     .                IDIMEN(2),HK0(2)   ,EMISIV   ,RADIAT   ,
     .                HCAPS
      LOGICAL         TRBLNT
      COMMON /HETDT4/ TRBLNT(20),REYNLD(20),ENTHWF(20),ENTHRF(20),
     .                TMPRF(20) ,ENSTAR(20),TSTAR(20) ,HTRANF(20),
     .                HIF(20)   ,HTF(20)
      COMMON /HETDT5/ XDISTN(20),RNREF(20) ,ALOGRT(20),COEFM(20),
     .                C0(20)    ,HCAPF     ,C5(20)    ,DANGL(20)
      COMMON /HTOUT1/ C1    ,C2(10),C3    ,C4(10),C6    ,C7    ,
     .                F1(10),F2(10),HK1   ,HK2
      LOGICAL         DSC
      COMMON /ICBOXD/ DSC(6)
      COMMON /IDXAC1/ NCA
      COMMON /IDXA31/ IAA       ,ICA
      COMMON /IDXBL1/ MCA
      COMMON /IDXTM1/ LAA       ,LCA       ,LCB       ,LCC       ,LCD       ,
     .                LEA       ,LEB       ,LEC       ,LED
      COMMON /IDXT21/ JCA
      COMMON /IDXT31/ JAA3      ,JAB3      ,JAC3      ,JAD3      ,JCA3      ,
     .                JCB3
      COMMON /INERTP/ VOI   ,ALPOI ,BTAOI ,GMAOI
      COMMON /INGDAT/ NEQN  ,HI    ,H2
      COMMON /INGDA3/ HI3
      COMMON /INTDAT/ IH    ,IPER  ,IERR  ,INTTMB,INTBMX,INTTME,
     .                INTEMX,INTTMU,INTUMX,INTCNT,INTTBA,INTTEA,
     .                INTTUA,INTRCL
      COMMON /IN3DAT/ IH3   ,IPER3 ,IERR3 ,IN3TMB,IN3BMX,IN3TME,
     .                IN3EMX,IN3TMU,IN3UMX,IN3CNT
      COMMON /JETDAT/ XJETF ,YJETF ,XJETP ,ZJETP ,FRCJET,QBARSW,
     .                JETC(12),JETI(12)
      CHARACTER*8     MCLABL
      COMMON /MCDEFL/ MCLABL
      CHARACTER*8     MCARRY
      COMMON /MCMENA/ INDXMC,MCARRY(96)
      COMMON /MCVALU/ PFUEL ,XMCMAP,YMCMAP
      CHARACTER*12    NAMVM
      CHARACTER*1     ITPVRM
      LOGICAL*1       DGMODM,WGHMDM,FOUNDM
      COMMON /MSCLIS/ VALUM(60),IADRM(60),NAMVM(60),DGMODM(60),
     .                WGHMDM(60),ITPVRM(60),NBYTSM(60),FOUNDM(60),
     .                MSCNUM
      CHARACTER*6     MCDEV
      LOGICAL         LMSCMP,MCRST ,MCLEAR
      COMMON /MSSCMP/ LMSCMP,MCRST ,MCLEAR,MCNT  ,MCNTMX,MCSTAT,
     .                MCDLAY,MCDEV
      COMMON /MOMNTS/ AL    ,AM    ,ANN
      LOGICAL         LRECV
      COMMON /MSSGES/ NUMMES,IRTADR(62),ISUBA(62),IWRDS(62),
     .                INDX(62),LRECV(62)
C##      EXTENDED BASE  /NASPSC/ 356
      CHARACTER       LSTNAM*8,LSTTYP*1
crc      INTEGER*1       LSTLEN
      INTEGER*2       LSTLEN    !dkh
C##      EXTENDED BLOCK
      COMMON
     .       /NASPSC/ LSTNAM(3500),LSTTYP(3500),
     .                LSTLEN(3500),LSTADR(3500)
C##      EXTENDED BASE  /NSPACA/ 268
C##      EXTENDED BLOCK
      COMMON
     .       /NSPACA/ CL0_2A (    189),CD0_2A (    189),CM0_2A (    189),
     .                CMDE_2A(    189),CMQ_2A (    189)
C##      EXTENDED BASE  /NSPAM3/ 292
C##      EXTENDED BLOCK /NSPAM3/
      COMMON
     .                CL0_3A (    375),CD0_3A (    375),CDI_3A (     34),
     .                CM0_3A (    375)
C##      EXTENDED BASE  /NSPBLA/ 272
C##      EXTENDED BLOCK
      COMMON
     .       /NSPBLA/ CL0_BA (    169),CD0_BA (    169),CM0_BA (    169),
     .                CMDE_BA(    169),CMQ_BA (    169)
C##      EXTENDED BASE  /NSPBLP/ 280
C##      EXTENDED BLOCK
      COMMON
     .       /NSPBLP/ C1ISPA (    810),C2ISPA (    450),C3ISPA (    540),
     .                C4ISPA (    540),CAPR1A (     90),CAPR2A (     50),
     .                CAPR3A (     54),CAPR4A (     54),CDEOA  (     17)
C##      EXTENDED BASE  /NSPDUM/ 256
C##      EXTENDED BLOCK
      COMMON
     .        /NSPDUM/ ZZZZZZ
C##      EXTENDED BASE  /NSPHET/ 276
C##      EXTENDED BLOCK
     .       /NSPHET/ ENTHA  (    192),C1A    (      7),C2A    (     48),
     .                C3A    (      7),C4A    (     48),C6A    (     32),
     .                C7A    (     32),F1A    (     49),F2A    (     49),
     .                HK1A   (      7),HK2A   (      7)
      COMMON /ORIGIC/ FIC(13)
      COMMON /OVPRES/ BK1   ,BK2   ,BK3   ,BODLEM,CROOTM,CSDIAM,
     .                OVPLNM,OVPRNM,OVPVNM,OVRPRS,PANM  ,PONM  ,
     .                XLIFTN,YM
      COMMON /PCSOUT/ P01OT1,P02OT1,P03OT1,P04OT1,P05JN1,P06OT1,
     .                P07OT1,P08OT1,P09JN1,P10JN1,P11OT1,P12OT1,
     .                P13OT1,P14OT1,P15JN1,P16OT1,P17OT1,P18OT1
      CHARACTER       CPEN*32,NAMVP*12,ITPVRP*1
      LOGICAL*1       DGMODP,WGHMDP,FOUNDP
      COMMON /PENLIS/ VALUP(12) ,IADRP(12) ,NAMVP(12) ,DGMODP(12),
     .                WGHMDP(12),ITPVRP(12),NBYTSP(12),FOUNDP(12),
     .                CPEN(12)
```

```
      COMMON /PFILTK/ P04F1K,P04F2K,P06F1K,P06F2K,P13F1K,P13F2K,
     .                P13F3K,P14F1K
      COMMON /PGAINS/ XKDEP ,XKP   ,XKM   ,XKQBAR,XKI
      COMMON /PHIDAT/ PHICA(20),PHIMA(20),NUMPHI
      LOGICAL         USEPIL,PILTIC
      COMMON /PILUSE/ USEPIL,PILTIC
      COMMON /PRINFO/ IR    ,OPI   ,TMAX  ,ITITL(12)
      COMMON /PTHGAN/ P16KAN,P17KAN,P18KAN,R08KAN,R09KAN,Y21KAN,
     .                Y22KAN,Y23KAN,Y24KAN,Y25KAN
      COMMON /QBGOUT/ PHISTB,XLIFTV,XLIFTE,HDOTRF,DLIFT ,XLIFTC,
     .                DTHAC ,QBK1  ,QBK2  ,QBK3  ,QBARRF,ZETA  ,
     .                QBARCA(10),QBARMA(10),NUMQBR
      COMMON /RATAC1/ CAM21 ,CALP9
      COMMON /RATAD1/ CAL25 ,CAM15 ,CAM34
      COMMON /RATBL1/ BALP9 ,BAM13
      COMMON /RATTM1/ TALP9 ,TAMA9 ,TAMB5 ,TAMC6 ,TAMD6 ,TAM17 ,TPH10
      COMMON /RATT21/ UAM17 ,UPH11
      COMMON /RATT31/ UAMC5 ,UAMC2 ,UAMC3 ,UAMC4 ,UAMC6 ,UALP9
      COMMON /RCSJET/ RCSN  ,RCSA  ,RCSY  ,RCSM  ,RCSYM ,RCSL
      COMMON /RCSOUT/ R01OT1,R02OT1,R03OT1,R04JN1,R05OT1,R06OT1,
     .                R07OT1,R08OT1,R09OT1
      REAL*8          REQUAT,RPOLE ,OMEGE ,RADIUS,RADIC ,RADRWY
      COMMON /REARTH/ REQUAT,RPOLE ,OMEGE ,RADIUS,RADIC ,RADRWY,
     .                GZERO ,SINLAT,COSLAT,TANLAT
      COMMON /REVNUM/ IREVNM
      COMMON /RGAINS/ XKDAP ,XKMA  ,XKQBRA
      LOGICAL         ONROCK
      COMMON /ROCKET/ ONROCK,EXPRAT,RKTHR0,XIMPLS,AREACN,AREATH,
     .                RKTHRS,RKFFLO
      LOGICAL         RSTREC,RSTTME
      COMMON /RSTTIM/ RSTREC,RSTTME
      LOGICAL         OP    ,RST   ,HLD   ,RT    ,ATRM  ,ICEN  ,MDAT
      COMMON /RTCDAT/ OP    ,RST   ,HLD   ,RT    ,ATRM  ,ICEN  ,MDAT
      CHARACTER*24    RDWPTH,RDRPTH,RDWPTL,RDRPTL
      LOGICAL         RTINIT,RTWRIT
      COMMON /RTDATA/ RTDATW(174),RTDATR(174),RTINIT,RTWRIT,
     .                NRUNW ,NRUNR ,IFRAMW,IFRAMR,IWRITS,IREADS,
     .                IWRTFR,IREDFR,RDWPTH,RDRPTH,RDWPTL,RDRPTL
C##         COMMON /RT:DIS/ ISSW(8),IOSW(8)
C##         COMMON /RT:INC/ INCERR
      COMMON /RTSTAT/ RT1RAT
      REAL*8          XLAT0 ,XLNG0 ,RNWLAT,RNWLNG
      COMMON /RUNWAY/ XLAT0 ,XLNG0 ,ROTRW ,SINHDR,COSHDR,XDISRW,
     .                YDISRW,XRNWAY,YRNWAY,ROTVIS,SINVIS,COSVIS,
     .                XVIS  ,YVIS  ,ROTMAP,SINMAP,COSMAP,XMAP  ,
     .                YMAP  ,PSIM  ,XVIS0 ,YVIS0 ,RNWLAT,RNWLNG
      LOGICAL         LPRNT ,FFEED ,LLOCK ,UNLOCK,STNDBY,LRUN  ,
     .                CHRTON,LMARK ,LSCALE,SCINIT
      CHARACTER*40    PENBUF
      COMMON /SCDATA/ LPRNT ,FFEED ,LLOCK ,UNLOCK,STNDBY,LRUN  .
     .                CHRTON,LMARK ,LSCALE,SCINIT,MMPSEC,MMPMIN,
     .                MMPHR ,IRUN  ,PENBUF(16)
      CHARACTER*8     SCLABL
      COMMON /SCDEFL/ SCLABL
      CHARACTER*8     SCARRY
      COMMON /SCMENA/ INDXSC,SCARRY(96)
      CHARACTER*2     SCTYPE
      COMMON /SCTYPE/ SCTYPE
      COMMON /SELDAT/ XACCEL, YACCEL, ZACCEL
      REAL*8          TIC   ,PIC   ,QIC   ,RIC   ,RUHIC ,PSIHIC,
     .                WHIC  ,THAIC ,PSIIC ,PHIIC ,DELRIC,XLATIC,
     .                XLNGIC
      COMMON /SETICS/ TIC   ,PIC   ,QIC   ,RIC   ,RUHIC ,PSIHIC,
     .                WHIC  ,THAIC ,PSIIC ,PHIIC ,DELRIC,XLATIC,
     .                XLNGIC
      COMMON /SETIC2/ VIC   ,ALPIC ,BTAIC ,HIC   ,SALPIC,CALPIC,
     .                SBTAIC,CBTAIC
      COMMON /SIGNAL/ SIGNL
      COMMON /SIMACC/ AXB   ,AYB   ,AZB   ,ANX   ,ANY   ,ANZ   ,AN
      CHARACTER*80    MACFIL,SCRFIL
      LOGICAL         LMACRO,SCRIPT
      COMMON /SIMCON/ MACFIL,SCRFIL,LMACRO,SCRIPT,LFCM  ,LFCS
      COMMON /SIMOUT/ AMCH  ,QBAR  ,GMA   ,DELFP ,UBRA  ,VBRA  ,
     .                WBRA  ,VEAS  ,VCAS
      LOGICAL         RUNSTP
      COMMON /SIMSTP/ RUNSTP,ALPMAX,ALPMIN,BTAMAX,NFRHLD,HMIN
      LOGICAL         NORMAL,LONGIT,LATDIR,LEVEL ,BYPASS,TYPEIN
      COMMON /SIMTYP/ NORMAL,LONGIT,LATDIR,LEVEL ,BYPASS,TYPEIN
      COMMON /SPEEDB/ SBDRAG,CDSB  ,CDSB1 ,SSB   ,PRCTSB
      REAL*8          THATK ,PHITK ,PSITK ,XTK   ,YTK   ,HTK
      LOGICAL         PYBM  ,FXRG
      COMMON /TARGET/ THATK ,PHITK ,PSITK ,XTK   ,YTK   ,HTK   ,
     .                VEASTK,RANGET,WUTG  ,REVT  ,PSICI ,F1E2  ,
     .                F1K2  ,PYBM  ,TARH  ,TARX  ,TARY  ,FXRG
crc      INTEGER*1       HOMEC ,CURSOR,ENABLC,BLANKC,BLANKL
      INTEGER*2       HOMEC ,CURSOR,ENABLC,BLANKC,BLANKL  !dkh
      LOGICAL         TWOTEL
      COMMON /TERMNL/ ITYCRT   ,TWOTEL   ,HOMEC(10) ,CURSOR(10),
     .                ENABLC(10),BLANKC(10),BLANKL(80),IHOME     ,
     .                ICURS    ,IENABL   ,IBLNKC    ,IBLNKL
      COMMON /TMOUT1/ C1ISP ,C2ISP ,C3ISP ,C4ISP ,CAPR1 ,CAPR2 ,CAPR3 ,
     .                CAPR4 ,CDEO
      COMMON /TRIGFN/ SINALP,COSALP,SINBTA,COSBTA,SINPHI,COSPHI,
     .                SINPSI,COSPSI,SINTHA,COSTHA
      COMMON /TRIMIN/ YTRIM
C##      EXTENDED BASE   /NSPACP/ 288
C##      EXTENDED BLOCK
      COMMON
     .       /NSPACP/ EISPAA (   187),CAPRAA (   187)
      COMMON /T2OUT1/ EISPA ,CAPRA
      COMMON /T3DAT1/ CTA1A  (     5),CTA2A  (     2),CTA3A  (     3),
     .                CTA4A  (    54),CTN1A  (     5),CTN2A  (     2),
     .                CTN3A  (     3),CTN4A  (     4),CAPR31A(     5),
     .                CAPR32A(     2),CAPR33A(     3),CAPR34A(    36),
     .                EISP31A(     5),EISP32A(     2),EISP33A(     3),
     .                EISP34A(     4)
      COMMON /T3OUT1/ CTA1  ,CTA2  ,CTA3  ,CTA4  ,CTN1  ,CTN2  ,CTN3  ,
     .                CTN4  ,CAPR31,CAPR32,CAPR33,CAPR34,EISP31,EISP32,
     .                EISP33,EISP34
      LOGICAL*1       LX
      COMMON /VARDAT/ UX(90),LX(50)
      LOGICAL*1       LS
      COMMON /VARSIM/ US(90),LS(50)
      COMMON /VSCDAT/ VBAR (22),TRATIO(22),TPRIME(22),CINF (22),
     .                REYINF   ,TINF     ,SHRAT
      LOGICAL         WIND
      COMMON /WINDAT/ WIND  ,XWIND ,YWIND ,NUMWND,ALTW(32),VELW(32),
     .                HDGW(32),XWA(32),YWA(32)
      COMMON /YCSOUT/ Y01OT1,Y02OT1,Y03OT1,Y04OT1,Y05OT1,Y06OT1,
     .                Y07JN1,Y08OT1,Y09OT1,Y10OT1,Y11OT1,Y12OT1,
     .                Y13OT1,Y14OT1,Y15OT1,Y16JN1,Y17JN1,Y18JN1,
     .                Y19OT1,Y20OT1,Y21OT1,Y22OT1,Y23OT1,Y24OT1,
     .                Y25OT1
      COMMON /YFILTK/ Y01F1K,Y05F1K,Y05F2K,Y05F3K,Y05F4K,Y05F5K,
     .                Y08F1K,Y08F2K,Y10F1K,Y10F2K,Y12F1K,Y12F2K
      COMMON /YGAINS/ XKRT  ,XKDRP ,XKR   ,XKMR  ,XKQBRR,XKRR
```

```
c.....NASPGCom.inc
C
C.... GRAM PROGRAM COMMON BLOCKS.   2/18/89   LJS.
C     SOME GRAM PARAMETER NAMES WERE CHANGED IN THIS ROUTINE
C     TO AVOID CONFLICT WITH DRYDEN SIM PARAMETER NAMES.
C     GRAM PARAMETER NAMES WITHIN THE GRAM ROUTINES WERE UNCHANGED.
C
C     EXTENDED'S REMOVED FOR MAC   10/16/89   LJS.
C
C##      EXTENDED BASE  /C4    / 464
C##      EXTENDED BLOCK /C4    /
      COMMON /C4    / GLAT(16),GLON(16),NG    ,P4D(16,26),D4D(16,26),
     .                T4D(16,26),SP4(16,26),SD4(16,26),ST4(16,26),
     .                THET1 ,THET  ,HS
      COMMON /CHIC  / LA(4,4),NB(2),IWSYM,UCOEF(14,9),VCOEF(14,9)
      COMMON /COMJAC/ XLATJ ,XLONG ,SDA   ,SHA   ,DY    ,R88   ,TE    ,
     .                EM
      COMMON /COMPER/ SPH   ,SDH   ,STH   ,PRH   ,DRH   ,TRH   ,URH   ,
     .                VRH   ,SUH   ,SVH   ,CP88  ,PRHS  ,DRHS  ,TRHS  ,
     .                URHS  ,VRHS  ,PRHL  ,DRHL  ,TRHL  ,URHL  ,VRHL  ,
     .                SPHS  ,SDHS  ,STHS  ,SUHS  ,SVHS  ,SPHL  ,SDHL  ,
     .                STHL  ,SUHL  ,SVHL
      COMMON /IOTEMP/ IOTEM1,IOTEM2,IUG   ,IUN   ,DD88  ,XMJD  ,PHI1  ,
     .                PHI88 ,NSAME ,RP1   ,RD1   ,RT1   ,SP1   ,SD1   ,
     .                ST1   ,RU1   ,RV1   ,SU1   ,SV1   ,MN    ,IDA88 ,
     .                IYR   ,H1    ,PHI1R ,THET1R,G88   ,RI    ,H88   ,
     .                PHIR  ,THETR ,F10   ,F10B  ,AP    ,IHR   ,MIN   ,
     .                NMORE ,DX    ,HL    ,VL    ,DZ    ,B88   ,EPS   ,
     .                IOPP  ,LOOK  ,IET   ,GLATX ,RP1S  ,RD1S  ,RT1S  ,
     .                RU1S  ,RV1S  ,SP1S  ,SD1S  ,ST1S  ,SU1S  ,SV1S  ,
     .                UDS1  ,VDS1  ,UDL1  ,VDL1  ,UDS2  ,VDS2  ,UDL2  ,
     .                VDL2, REARTH
      COMMON /IPRTP / IPRT
C##      EXTENDED BASE  /PDTCOM/ 448
C##      EXTENDED BLOCK /PDTCOM/
      COMMON /PDTCOM/ IU4   ,MONTH ,IOPR  ,PG88(18,19),TG(18,19),
     .                DG(18,19),PSP(8,10,12),DSP(8,10,12),TSP(8,10,12),
     .                PAQ(17,5),DAQ(17,5),TAQ(17,5),PDQ(17,5),DDQ(17,5),
     .                TDQ(17,5),PR(20,10),DR88(20,10),TR(20,10),
     .                UAQ(17,5),VAQ(17,5),UDQ(17,5),VDQ(17,5),UR(25,10),
     .                VR(25,10),PQ    ,DQ    ,TQ    ,UQ    ,VQ88  ,
     .                PQA   ,DQA   ,TQA   ,UA    ,VA    ,IOPQ  ,
     .                PLP(25,10),DLP(25,10),TLP(25,10),ULP(25,10),
     .                VLP(25,10),UDL(25,10),VDLA(25,10),UDS(25,10),
     .                VDSA(25,10)
      COMMON /WINCOM/ DH    ,FCORY ,DX5   ,DY5   ,DPX   ,DPY   ,DPXX  ,
     .                DPXY  ,DPYY  ,UGH   ,VGH   ,TH    ,DTX   ,DTY   ,
     .                DUH   ,DVH   ,PH    ,UPRE  ,VPRE  ,DUPRE ,DVPRE
C
C.... COMMON BLOCKS ADDED IN MODIFYING GRAM AND INTERFACING WITH SIM.
C
      COMMON /GRAMOT/ PGH   ,DGH   ,TGH   ,UH    ,VH    ,PS    ,DS    ,
     .                TS    ,PGHP  ,DGHP  ,TGHP  ,PHP   ,DHP   ,THP   ,
     .                PSH   ,DSH   ,TSH   ,WGH
      LOGICAL         GRMATM,G76ATM,GATMP ,GRMWND,GWINDP
      COMMON /GRMDAT/ GRMATM,G76ATM,GATMP ,GRMWND,GWINDP,CS76  ,CSU   ,
     .                CSP   ,TMPP76,TMPPU ,TMPPP ,PA76  ,PAU   ,PAP   ,
     .                RHO76 ,RHOU  ,RHOP  ,UWINDU,UWINDP,VWINDU,VWINDP,
     .                USHEAR,VSHEAR
C##      EXTENDED BASE  /NASPGM/ 380
C##      EXTENDED BLOCK /NASPGM/ PDAT(5720) ,DDAT(5720) ,TDAT(5720) ,
      COMMON /NASPGM/ PDAT(5720) ,DDAT(5720) ,TDAT(5720) ,
     .                SPDAT(5720),SDDAT(5720),STDAT(5720)
```

```
c.....OptFlg.inc

c.....user option flags

      integer*2                oCycle     ! cycle through event loop
      integer*2                oSave      ! save Map as Pict file
      integer*2                oRedraw    ! redraw Map
      integer*2                oNew       ! make new Map
      integer*2                oQuit      ! quit program

      common / OptFlg /        oCycle,      oSave,       oRedraw,     oNew,
     &                         oQuit
```

```
c.....PenCom.inc

c     common block containing array of pen commands

      common / PenCom /                         PenCommand

      integer*1                PenCommand(13120)
```

```
c.....PicGrp.inc

c.....Pict group flags

      common / PicGrp / PicGroupBeg,PicGroupEnd

      integer*2         PicGroupBeg
      integer*2         PicGroupEnd
```

```
c.....PlotMenu.inc

c.....Plot menu declarations

      integer*2               nPlotItems
      integer*2               PlotItemNewPlot
      integer*2               PlotItemSavePlot
      integer*2               PlotItemRedrawPlot
      integer*2               PlotMenuID
      record / MenuHandle /   PlotMenuHndl

c.....Plot menu common block

      common / PlotMenu /   PlotItemNewPlot,        PlotItemSavePlot,
     .                      PlotItemRedrawPlot,
     .                      PlotMenuID,
     .                      PlotMenuHndl

c.....Plot menu pre-settings

      parameter            ( nPlotItems           =   3 )
      parameter            ( PlotItemNewPlot      =   1 )
      parameter            ( PlotItemSavePlot     =   2 )
      parameter            ( PlotItemRedrawPlot   =   3 )
      parameter            ( PlotMenuID           = 131 )
```

```
c.....pntabs.inc

```

```
      COMMON / PNTABS /
     .  IXABS          , IYABS

      INTEGER   IXABS
      INTEGER   IYABS
```

```
c.....RunSetup.inc

c.....dialog declarations

      integer*2                rRunSetupDLOG
      integer*2                rCloseButton
      integer*2                rRunButton
      integer*2                rMapButton
      integer*2                rSaveButton
      integer*2                rDegWestButton
      integer*2                rDegEastButton
      integer*2                rMeters
      integer*2                rKilometers
      integer*2                rSeconds
      integer*2                rMinutes
      integer*2                rHours
      integer*2                rLatitude
      integer*2                rLongitude
      integer*2                rAltitude
      integer*2                rDuration
      integer*2                rMissionLabel
      integer*2                rWindModelSelector
      integer*2                rWindModelSelectorPopup
      integer*2                rAscentSelector
      integer*2                rAscentSelectorPopup

c.....item ids for dialog-related saved resources

      integer*2                rOldMissionText
      integer*2                rOldLatitude
      integer*2                rOldLongitude
      integer*2                rOldDuration
      integer*2                rOldAltitude
      integer*2                rOldAscent
      integer*2                rOldClimate
      integer*2                rOldDegRadio
      integer*2                rOldDistRadio
      integer*2                rOldTimeRadio

c.....working variable declarations

      integer*2                DType
      integer*2                ItemHit
      integer*2                rDegreeSelection
      integer*2                rDistanceSelection
      integer*2                rTimeSelection
      integer*2                ClimateSelection
      character*255            AscentSelection

      real                     xLatitude
      real                     xLongitude
      real                     xAltitude
      real                     xDuration
      character*255            xMissionLabel

c.....dialog pre-settings

```

```
      parameter          ( rRunSetupDLOG          = 256 )
      parameter          ( rCloseButton           =   1 )
      parameter          ( rRunButton             =   2 )
      parameter          ( rMapButton             =   3 )
      parameter          ( rSaveButton            =   4 )
      parameter          ( rDegWestButton         =   5 )
      parameter          ( rDegEastButton         =   6 )
      parameter          ( rMeters                =   7 )
      parameter          ( rKilometers            =   8 )
      parameter          ( rSeconds               =   9 )
      parameter          ( rMinutes               =  10 )
      parameter          ( rHours                 =  11 )
      parameter          ( rMissionLabel          =  19 )
      parameter          ( rLatitude              =  20 )
      parameter          ( rLongitude             =  21 )
      parameter          ( rAltitude              =  22 )
      parameter          ( rDuration              =  23 )
      parameter          ( rWindModelSelector     =  24 )
      parameter          ( rWindModelSelectorPopup =  41 )
      parameter          ( rAscentSelector        =  25 )
      parameter          ( rAscentSelectorPopup   =  43 )

      parameter          ( rOldMissionText        = 1000 )
      parameter          ( rOldLatitude           = 1001 )
      parameter          ( rOldLongitude          = 1002 )
      parameter          ( rOldDuration           = 1003 )
      parameter          ( rOldAltitude           = 1004 )
      parameter          ( rOldAscent             = 1011 )
      parameter          ( rOldClimate            = 1012 )
      parameter          ( rOldDegRadio           = 1021 )
      parameter          ( rOldDistRadio          = 1022 )
      parameter          ( rOldTimeRadio          = 1023 )


c.....dialog-related structures

      record / GrafPort      /  SavedPort
      record / DialogPtr     /  GetSelection
      record / DialogPeek    /  TheDialogPtr
      record / TEHandle      /  ThisEditText
      record / Handle        /  DItem
      record / ControlHandle /  CItem
      record / Rect          /  tempRect

c.....dialog-related common block

      common / RunSetup /
     &                          SavedPort,
     &                          GetSelection,
     &                          DType,
     &                          DItem,
     &                          tempRect,
     &                          rDegreeSelection,
     &                          rDistanceSelection,
     &                          rTimeSelection

c.....trajectory input common block

      common / TrajInput
     &                        xLatitude,
     &                        xLongitude,
     &                        xAltitude,
     &                        xDuration,
     &                        xMissionLabel,
     &                         ClimateSelection,
     &                         AscentSelection
```

```
c.....TicDat.inc

c.....set maximum number major and minor tic marks

      parameter        ( nticmx = 20 )

c.....common storage for items relating to tic marks

      common / TicDat / xticmj,                xticmi,
     .                  yticmj,                yticmi,
     .                  lticmj,                lticmi,
     .                  ndivmj,
     .                  xrefmj,                yrefmj,
     .                  hticmj,                vticmj,
     .                  hticmi,                vticmi

c.....variable type declarations

      integer*2         xticmj
      integer*2         xticmi
      integer*2         yticmj
      integer*2         yticmi
      integer*2         lticmj
      integer*2         lticmi
      integer*2         ndivmj
      integer*4         nticmx
      real*4            xrefmj(nticmx)
      real*4            yrefmj(nticmx)
      integer*2         hticmj(nticmx)
      integer*2         vticmj(nticmx)
      integer*2         hticmi(nticmx,nticmx)
      integer*2         vticmi(nticmx,nticmx)
```

```
c.....traj.inc

c.....trajectory variables

      real*4 Time_Array
      real*4 LAT_ARRAY
      real*4 LON_ARRAY
      real*4 ALT_ARRAY
      real*4 GRANGE_ARRAY
      real*4 WINDAZ_ARRAY
      real*4 WIND_VEL_ARRAY

      integer*4 No_Of_Pts

c.....trajectory variable common block

      common / traj /
     &                     Time_Array( 2048 ),
     &                     LAT_ARRAY ( 2048 ),
     &                     LON_ARRAY ( 2048 ),
     &                     ALT_ARRAY ( 2048 ),
     &                     GRANGE_ARRAY ( 2048 ),
     &                     WINDAZ_ARRAY ( 2048 ),
     &                     WIND_VEL_ARRAY ( 2048 ),
     &                     No_Of_Pts
```

```
c.....TrjCom.inc

c.....trajectory data common block

      integer*2       MaxPts
      parameter       (MaxPts=2048)

      real*4          TofTab    (MaxPts)
      real*4          LngTab    (MaxPts)
      real*4          LatTab    (MaxPts)
      real*4          AltTab    (MaxPts)

      integer*2       JmpTab    (MaxPts)
      integer*2       ntrpts

      common / TrjCom / TofTab,     LngTab,     LatTab,     AltTab,
     &                  JmpTab,     ntrpts
```

```
c.....TrjLim.inc

c.....trajectory data limits

      common / TrjLim / MinTof,     MaxTof,     MinLng,     MaxLng,
     &                  MinLat,     MaxLat,     MinAlt,     MaxAlt

      real*4          MinTof
      real*4          MaxTof
      real*4          MinLng
      real*4          MaxLng
      real*4          MinLat
      real*4          MaxLat
      real*4          MinAlt
      real*4          MaxAlt
```

```
c.....VuWind.inc

c.....view window records

      common / VuWind / ScrollHndl, TxHndl,     FntDat,     viewRect,
     .                  destRect,   TxWptr,     iVuPag,     ScrollPart

c.....scroll bar handle

      record / ControlHandle /    ScrollHndl

c.....text edit handle

      record / TEHandle /         TxHndl

c.....font characteristics

      record / FontInfo /         FntDat

c.....Rectangle records

      record / rect /             viewRect
      record / rect /             destRect

c.....text display window records
```

```

      record / WindowPtr /       TxWptr

c.....view window page size

      integer*2                  iVuPag

c.....scroll part index

      integer*2                  ScrollPart
```

```
c.....winlim.inc

c     graphics and Map window limits

      common / winlim /
     .  igxmin,     igxmax,     igymin,     igymax,
     .  ipxmin,     ipxmax,     ipymin,     ipymax,
     .  iwxmin,     iwxmax,     iwymin,     iwymax

      integer*2     igxmin
      integer*2     igxmax
      integer*2     igymin
      integer*2     igymax
      integer*2     ipxmin
      integer*2     ipxmax
      integer*2     ipymin
      integer*2     ipymax
      integer*2     iwxmin
      integer*2     iwxmax
      integer*2     iwymin
      integer*2     iwymax
```

10.3 BDPS FORTRAN SOURCE CODE

This section contains a complete listing of the FORTRAN source code which was used in the generation of the BDPS executable program. The source files lines are numbered by the FORTRAN compiler. The majority of the files exist to implement the Macintosh graphical interface and the drift pattern display. Two files, "RunTraj.f" and "Gen4d.f," contain a version of NASA's GRAM which was modified for BDPS. The option for using external forecast data files makes use of the "NOGAPS.f" source file.

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main

c----------------------------------------------------------------------------
```

```
Segment Main
      subroutine AlertUser
c-------------------------------------------------------------------------------
c     Display an alert that tells the user an error occurred, then
c     exit the program.

!!SETC USINGINCLUDES = FALSE
      implicit none

c.....set up pointer for QuickDraw globals

      pointer / QDGlobals /       qdg
      common  / QDGPtr /          qdg

      integer*2   rUserAlert
      integer*2   itemHit

      parameter ( rUserAlert=129 )

c-------------------------------------------------------------------------------

      call SetCursor(qdg^.Arrow)
      itemHit = Alert(rUserAlert, nil)
      call ExitToShell

      return
      end
```

```
c-------------------------------------------------------------------------------
      subroutine AutoScale ( xmin , xmax , ndivmj , dxmin , dxmax ,
     .                       xdivmj , xdivmi )
c-------------------------------------------------------------------------------
c     Compute plot extremes [dxmin] and [dxmax] which will enclose the data
c     extremes [xmin] and [xmax] and yield [ndivmj] divisions [xdivmj] wide,
c     each composed of minor divisions [xdivmi] wide.  The major and minor
c     divisions have only one significant figure each.

c.....external function declaration

      external        nquant
      integer*2     nquant

c.....if data extremes are equivalent, handle as a special case

      if ( xmin.eq.xmax ) then
         if ( xmin.ne.0.0 ) then
            ilogx  = iint ( alog10(abs(xmin)) - 1.0 )
         else
            ilogx  = 0
         end if
         xdivmj = 10.0**ilogx
         xdivmi = xdivmj/10.0
         dxmin  = xmin - xdivmj
         dxmax  = xmin + xdivmj
         return
      end if

c.....major division width

      tmp1   = ( xmax - xmin )/float ( ndivmj )
      tmp2   = 10.0**nquant ( alog10(tmp1) , -1.0 )
      tmp3   = tmp1/tmp2
      if ( tmp3.gt.anint(tmp3) ) then
```

```
            xdivmj = tmp2*anint ( tmp3 + 1.0 )
         else
            xdivmj = tmp2*anint ( tmp3 )
         end if

c.....minor division width

         xdivmi = 10.0**nquant ( alog10(xdivmj) , -1.0 )
         if ( xdivmi.eq.xdivmj ) then
            xdivmi = xdivmj/10.0
         end if

c.....compute the width of the plot window and the span of the input data

         xwidth = xdivmj*float ( ndivmj )
         xspan  = xmax - xmin

c.....compute excess width provided with respect to the span of the data
c        extremes and quantize it to the nearest minor division

         xces   = xwidth - xspan
         nxces  = nquant ( xces , -xdivmi )

c.....allocate half of the excess to the lower end of the plot scale and
c        quantize the minimum plot value to the nearest minor division

         nxmin  = nquant ( xmin , -xdivmi )
         dxmin  = xdivmi*float ( nxmin - nxces/2 )

c.....offset the maximum plot value from the minimum plot value by the width
c        of the plot window

         dxmax  = dxmin + xwidth

         return
         end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G ToolBox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main

c-----------------------------------------------------------------------------------
Segment Main
         program bdps
c-----------------------------------------------------------------------------------
c        Compute and display balloon drift patterns:
c        specifically configured for a Phase 2 SBIP contract with DARPA.
c
c        Developer:     Robert L. Hawkins
c        Date:               May 1991
c        Dialect:       Language Systems FORTRAN 2.1, MPW 3.1
c        Machine:       Macintosh SE/30, 5mb RAM

c....Declare compile-time variable

```

```
!!SETC USINGINCLUDES = FALSE

cc      implicit none

c.....common block definition files

      include 'Alert.inc'
      include 'AppleMenu.inc'
      include 'EditMenu.inc'
      include 'FileMenu.inc'
      include 'Globals.inc'
      include 'MBar.inc'
      include 'naspcom.inc'
      include 'naspgcom.inc'
c      include 'PlotMenu.inc'
      include 'RunSetup.inc'
      include 'Traj.inc'

      include 'CrvDat.inc'
      include 'FntCom.inc'
      include 'LatCom.inc'
      include 'LngCom.inc'
      include 'MapMenu.inc'
      include 'OptFlg.inc'
      include 'PenCom.inc'
      include 'MapCom.inc'
      include 'PntAbs.inc'
      include 'TicDat.inc'
      include 'TrjCom.inc'
      include 'VuWind.inc'
      include 'WinLim.inc'

c.....set up pointer for QuickDraw globals

      pointer / QDGlobals /       qdg
      common  / QDGPtr /          qdg
      integer*4                   jQDGlobals
      external                    jQDGlobals

c-------------------------------------------------------------------------------

c.....give us room for memory allocation

      call MaxApplZone

c.....call Initialize then unload its segment from memory

      call Initialize
      call UnloadSeg( %loc(Initialize) )

c.....set up the pointer for QuickDraw globals

      qdg  = jQDGlobals()

c.....call eventloop; we will loop forever until user decides to quit

      call EventLoop

      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc
```

```

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main
c------------------------------------------------------------------------------
Segment Main
      subroutine ClearDegreeGroup
c------------------------------------------------------------------------------

!!SETC USINGINCLUDES = FALSE
      implicit none

c.....common block definition files

      include 'RunSetup.inc'

c------------------------------------------------------------------------------

c.....clear the degrees-west button

      call GetDItem( %val(GetSelection), %val(rDegWestButton),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )
      CItem.CtlH = DItem.bhdl
      call SetCtlValue( %val(CItem), %val(0) )

c.....clear the degrees-east button

      call GetDItem( %val(GetSelection), %val(rDegEastButton),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )
      CItem.CtlH = DItem.bhdl
      call SetCtlValue( %val(CItem), %val(0) )

      return
      end


c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main
c------------------------------------------------------------------------------
Segment Main
      subroutine ClearDistanceGroup
c------------------------------------------------------------------------------

!!SETC USINGINCLUDES = FALSE
      implicit none

c.....common block definition files

      include 'RunSetup.inc'

c------------------------------------------------------------------------------
```

```

c.....clear the meters radio button

      call GetDItem( %val(GetSelection), %val(rMeters),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )
      CItem.CtlH = DItem.bhdl
      call SetCtlValue( %val(CItem), %val(0) )

c.....clear the kilometers radio button

      call GetDItem( %val(GetSelection), %val(rKilometers),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )
      CItem.CtlH = DItem.bhdl
      call SetCtlValue( %val(CItem), %val(0) )

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main
c-----------------------------------------------------------------------------
Segment Main
      subroutine ClearTimeGroup
c-----------------------------------------------------------------------------

!!SETC USINGINCLUDES = FALSE
      implicit none

c.....common block definition files

      include 'RunSetup.inc'

c-----------------------------------------------------------------------------

c.....clear the seconds radio button

      call GetDItem( %val(GetSelection), %val(rSeconds),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )
      CItem.CtlH = DItem.bhdl
      call SetCtlValue( %val(CItem), %val(0) )

c.....clear the minutes radio button

      call GetDItem( %val(GetSelection), %val(rMinutes),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )
      CItem.CtlH = DItem.bhdl
      call SetCtlValue( %val(CItem), %val(0) )

c.....clear the hours radio button

      call GetDItem( %val(GetSelection), %val(rHours),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )
      CItem.CtlH = DItem.bhdl
      call SetCtlValue( %val(CItem), %val(0) )

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main
c-------------------------------------------------------------------------------
Segment Main
      integer*2 function CollectRunInput()
c-------------------------------------------------------------------------------
c     Collect all the values entered by the user in the Run Setup dialog

!!SETC USINGINCLUDES = FALSE
      implicit none

c.....common block definition files

      include 'Globals.inc'
      include 'RunSetup.inc'

c.....intermediate text string

      string*255 ItemText

c.....character intermediate

      character*255 CharData

c-------------------------------------------------------------------------------

c.....set the initial function value to zero (no problems)

      CollectRunInput = 0

c.....get the initial latitude in degrees north

      call GetDItem( %val(GetSelection), %val(rLatitude),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )
      call GetIText ( %val(DItem) , %val(ItemText) )
      CharData = ItemText
      read(CharData,*) xLatitude

c.....check to see that latitude value is ok

      if ( xLatitude.lt.-90.0 .or. xLatitude.gt.90.0 ) then
        CollectRunInput = rLatitude
        return
      endif

c.....get the initial longitude in degrees west

      call GetDItem( %val(GetSelection), %val(rLongitude),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )
      call GetIText ( %val(DItem) , %val(ItemText) )
      CharData = ItemText
      read(CharData,*) xLongitude

c.....check to see that longitude value is ok

      if ( xLongitude.lt.-180.0 .or. xLongitude.gt.180.0 ) then
        CollectRunInput = rLongitude
        return
      endif

c.....convert the longitude value if necessary

      if ( rDegreeSelection.eq.rDegWestButton ) then
        continue
      else if ( rDegreeSelection.eq.rDegEastButton ) then
        xLongitude = 360.0 - xLongitude
      endif

c.....get the initial altitude in meters

      call GetDItem( %val(GetSelection), %val(rAltitude),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )
      call GetIText ( %val(DItem) , %val(ItemText) )
      CharData = ItemText
      read(CharData,*) xAltitude

c.....check to see that altitude value is ok

      if ( xAltitude.lt.0.0 .or. xAltitude.gt.1000000.0 ) then
        CollectRunInput = rAltitude
        return
      endif

c.....convert the altitude value if necessary

      if ( rDistanceSelection.eq.rMeters ) then
        continue
      else if ( rDistanceSelection.eq.rKilometers ) then
        xAltitude = xAltitude * 1000.0
      endif

c.....get the flight duration in seconds

      call GetDItem( %val(GetSelection), %val(rDuration),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )
      call GetIText ( %val(DItem) , %val(ItemText) )
      CharData = ItemText
      read(CharData,*) xDuration

c.....check to see that flight duration value is ok (no more than 30 days)

      if ( xDuration.lt.0.0 .or. xDuration.gt.2592000.0 ) then
        CollectRunInput = rDuration
        return
      endif

c.....convert the flight duration value if necessary

      if ( rTimeSelection.eq.rSeconds ) then
        continue
      else if ( rTimeSelection.eq.rMinutes ) then
        xDuration = xDuration * 60.0
      else if ( rTimeSelection.eq.rHours ) then
        xDuration = xDuration * 3600.0
        endif

c.....get the mission label

      call GetDItem( %val(GetSelection), %val(rMissionLabel),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )
      call GetIText ( %val(DItem) , %val(ItemText) )
      CharData = ItemText

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbcx2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-------------------------------------------------------------------------------
      subroutine DashIt ( ixold , iyold , ixnew , iynew , icurve )
c-------------------------------------------------------------------------------
c     draw a dashed line between the end points.
c     note:      visible   Ÿ dark  Ÿ pen down ==> iPen = 1
c                invisible Ÿ white Ÿ pen up   ==> iPen = 0

      include 'CrvDat.inc'

c.....get Cartesian length components of line segment

      ixdif  = ixnew - ixold
      iydif  = iynew - iyold

c.....return if segment has zero length

      if ( iabs(ixdif).eq.0 .and. iabs(iydif).eq.0 ) then
         return
      end if

c.....compute unit vector parallel to line segment

      r      = sqrt  ( float(ixdif)**2 + float(iydif)**2 )
      ux     = float(ixdif)/r
      uy     = float(iydif)/r

c.....horizontal component exceeds vertical component

      if ( iabs(ixdif).ge.iabs(iydif) ) then

c........set loop indices

         ix1    = 0
         ix2    = iabs(ixdif)

c........loop on vertical scan lines

         do i = ix1 , ix2

c..........compute current pixel location

            ixref  = ixold + inint ( ux*float(i)/abs(ux) )
            iyref  = iyold + inint ( uy*float(i)/abs(ux) )
```

```

c...........increment the pixel counter for the current curve

            if ( ixref.ne.ixlast .or. iyref.ne.iylast ) then
               PixCnt(icurve) = 1 + imod ( PixCnt(icurve) , 16 )
               iBitNo         = 16 - PixCnt(icurve)
            end if

c...........see if current pixel is visible ( dark ) or invisible ( white )

            iPenLst = iPen
            iPen    = iibits ( DshMsk(LinTyp(icurve)) , iBitNo , 1 )

c...........save pen down location and draw as necessary

            if ( i.eq.ix1 ) then
               if ( iPen.eq.1 ) then
                  ixPenU = ixref
                  iyPenU = iyref
                  ixPenD = ixref
                  iyPenD = iyref
               end if
            else if ( i.gt.ix1 .and. i.lt.ix2 ) then
               if ( iPen.eq.1 ) then
                  ixPenU = ixref
                  iyPenU = iyref
                  if ( iPenLst.eq.0 ) then
                     ixPenD = ixref
                     iyPenD = iyref
                  end if
               else if ( iPen.eq.0 .and. iPenLst.eq.1 ) then
                  call MovAbs ( ixPenD , iyPenD )
                  call DrwAbs ( ixPenU , iyPenU )
               end if
            else if ( i.eq.ix2 ) then
               if ( iPen.eq.1 )  then
                  if ( iPenLst.eq.0 ) then
                     call MovAbs ( ixref   , iyref   )
                     call DrwAbs ( ixref   , iyref   )
                  else if ( iPenLst.eq.1 ) then
                     call MovAbs ( ixPenD , iyPenD )
                     call DrwAbs ( ixref   , iyref   )
                  end if
               end if
            end if

         end do

      end if

c.....vertical component exceeds horizontal component

      if ( iabs(ixdif).lt.iabs(iydif) ) then

c........set loop indices

         iy1    = 0
         iy2    = iabs(iydif)

c........loop on horizontal scan lines

         do i = iy1 , iy2

c...........compute current pixel location
```

```

            ixref  = ixold + inint ( ux*float(i)/abs(uy) )
            iyref  = iyold + inint ( uy*float(i)/abs(uy) )

c...........increment the pixel counter for the current curve

            if ( ixref.ne.ixlast .or. iyref.ne.iylast ) then
               PixCnt(icurve) = 1 + imod ( PixCnt(icurve) , 16 )
               iBitNo         = 16 - PixCnt(icurve)
            end if

c...........see if current pixel is visible ( dark ) or invisible ( white )

            iPenLst = iPen
            iPen    = iibits ( DshMsk(LinTyp(icurve)) , iBitNo , 1 )

c...........save pen down location and draw as necessary

            if ( i.eq.iy1 ) then
               if ( iPen.eq.1 ) then
                  ixPenU = ixref
                  iyPenU = iyref
                  ixPenD = ixref
                  iyPenD = iyref
               end if
            else if ( i.gt.iy1 .and. i.lt.iy2 ) then
               if ( iPen.eq.1 ) then
                  ixPenU = ixref
                  iyPenU = iyref
                  if ( iPenLst.eq.0 ) then
                     ixPenD = ixref
                     iyPenD = iyref
                  end if
               else if ( iPen.eq.0 .and. iPenLst.eq.1 ) then
                  call MovAbs ( ixPenD , iyPenD )
                  call DrwAbs ( ixPenU , iyPenU )
               end if
            else if ( i.eq.iy2 ) then
               if ( iPen.eq.1 )  then
                  if ( iPenLst.eq.0 ) then
                     call MovAbs ( ixref  , iyref  )
                     call DrwAbs ( ixref  , iyref  )
                  else if ( iPenLst.eq.1 ) then
                     call MovAbs ( ixPenD , iyPenD )
                     call DrwAbs ( ixref  , iyref  )
                  end if
               end if
            end if

         end do

      end if

c.....save Map end point for reference

      ixlast = ixnew
      iylast = iynew

      return
      end


c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

```

```
!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-----------------------------------------------------------------------
      subroutine DisplayMapLimits
c-----------------------------------------------------------------------
c     Display the text items containing the map limits

c.....include common block definition files

      include 'CrvDat.inc'
      include 'DefLim.inc'
      include 'FntCom.inc'
      include 'MapCom.inc'
      include 'MapLim.inc'
      include 'TrjLim.inc'

c.....item stuff

      record / handle /        ItHndl
      record / rect /          ItRect
      integer*4                ItType
      integer*2                ItNmbr
      string*255               ItText

c....."get Map data" dialog interface records

      common / MapSetUp  /     MapSetUpPtr,     iGotMapSetUp
      record / DialogPtr /     MapSetUpPtr
      integer*2                iGotMapSetUp

c.....character strings

      character*255            ChrDat

c.....set and select minimum latitude value

      ItNmbr = 5
      write(ChrDat,*) yMapMn
      ItText = ChrDat
      call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call SetIText ( %val(ItHndl) , %val(ItText) )
      call SelIText ( %val(MapSetUpPtr) , %val(ItNmbr) , %val(0) , %val(32767) )

c.....set maximum latitude value

      ItNmbr = 6
      write(ChrDat,*) yMapMx
      ItText = ChrDat
      call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call SetIText ( %val(ItHndl) , %val(ItText) )

c.....set latitude major division size

      ItNmbr = 7
      write(ChrDat,*) yDivMj
      ItText = ChrDat
      call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
```

```
      call SetIText ( %val(ItHndl) , %val(ItText) )

c.....set latitude minor division size

      ItNmbr = 8
      write(ChrDat,*) yDivMi
      ItText = ChrDat
      call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call SetIText ( %val(ItHndl) , %val(ItText) )

c.....set minimum longitude value

      ItNmbr = 9
      write(ChrDat,*) xMapMn
      ItText = ChrDat
      call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call SetIText ( %val(ItHndl) , %val(ItText) )

c.....set maximum longitude value

      ItNmbr = 10
      write(ChrDat,*) xMapMx
      ItText = ChrDat
      call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call SetIText ( %val(ItHndl) , %val(ItText) )

c.....set longitude major division size

      ItNmbr = 11
      write(ChrDat,*) xDivMj
      ItText = ChrDat
      call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call SetIText ( %val(ItHndl) , %val(ItText) )

c.....set longitude minor division size

      ItNmbr = 12
      write(ChrDat,*) xDivMi
      ItText = ChrDat
      call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call SetIText ( %val(ItHndl) , %val(ItText) )

c.....set time tic increment

      if ( TimeTics.eq.1 ) then
         write(ChrDat,*) tDivMj
      else
         ChrDat = ' '
      end if
      ItNmbr = 13
      ItText = ChrDat
      call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call SetIText ( %val(ItHndl) , %val(ItText) )

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c--------------------------------------------------------------------------------
      subroutine DisplayTimeTics ( xscale , xshift , yscale , yshift )
c--------------------------------------------------------------------------------
c     This subroutine displays time tic marks on the flight path.
c
c     developer:    David F. Smith
c     date:         February 1991

      include 'CrvDat.inc'
      include 'FntCom.inc'
      include 'LatCom.inc'
      include 'LngCom.inc'
      include 'MapCom.inc'
      include 'MapMenu.inc'
      include 'MapLim.inc'
      include 'OptFlg.inc'
      include 'PenCom.inc'
      include 'PicGrp.inc'
      include 'PntAbs.inc'
      include 'TicDat.inc'
      include 'TrjCom.inc'
      include 'TrjLim.inc'
      include 'VuWind.inc'
      include 'WinLim.inc'

c.....Rectangle records

      record / rect /           bounds

c.....Picture record handle and pointer

      common / pict   /         PictHndl
      record / PicHandle /      PictHndl
      record / PicPtr /         PictPtr

c.....set up pointer for QuickDraw globals

      common  / QDGPtr /        QDG
      pointer / QDGlobals /     QDG

c.....external function declaration

      external                  ntrvl
      integer*2                 ntrvl

c.....graduation output arrays

      parameter               ( mxGrad = 101 )
      integer*2                 mxGrad
      real*4                    grVal  (mxGrad)
      real*4                    pxRef  (mxGrad)
      real*4                    pyRef  (mxGrad)
      real*4                    uxRef  (mxGrad)
      real*4                    uyRef  (mxGrad)

c.....Intermediate text string
```

```

      string*255               TxtOut

c.....character strings

      character*255            ChrDat

c.....dialog interface variables ( note that pointers are i*4 )

      integer*4                nil

c-------------------------------------------------------------------------------
c--------------------------- graduate the plot curve ---------------------------
c-------------------------------------------------------------------------------

c.....calculate graduation values

      grVal(1) = tDivMj*anint ( tMapMn/tDivMj )
      do while ( grVal(1).lt.tMapMn .or. grVal(1).lt.MinTof )
         grVal(1) = tDivMj*anint ( 1.0 + grVal(1)/tDivMj )
      end do

      i      = 1
      do while ( grVal(i).le.tMapMx .and. grVal(i).le.MaxTof )
         grVal(i+1) = tDivMj*anint ( 1.0 + grVal(i)/tDivMj )
         i       = i + 1
      end do
      nGrads  = i - 1

c... .draw a tic mark for each graduation value

      do i = 1 , nGrads

c.......locate the data segment containing the graduation value

         j      = ntrvl ( grVal(i) , TofTab , ntrpts , 1 )
         k      = j + 1

c.......calculate latitude and longitude associated with the time point

         tmp1  = grval(i)  - TofTab(j)
         tmp2  = TofTab(k) - TofTab(j)
         yLat  = LatTab(j) + tmp1*( LatTab(k) - LatTab(j) )/tmp2
         if ( JmpTab(k).lt.0 ) then
            xLng   = LngTab(j) + tmp1*( LngTab(k) - 360.0 - LngTab(j) )/tmp2
            if ( xLng+180.0.lt.0.0 ) then
               xLng   = xLng + 360.0
            end if
         else if ( JmpTab(k).eq.0 ) then
            xLng   = LngTab(j) + tmp1*( LngTab(k) - LngTab(j) )/tmp2
         else if ( JmpTab(k).gt.0 ) then
            xLng   = LngTab(j) + tmp1*( LngTab(k) + 360.0 - LngTab(j) )/tmp2
            if ( xLng-180.0.gt.0.0 ) then
               xLng   = xLng - 360.0
            end if
         end if

c.......compute the window coordinates of the last data point

         pxLast = xshift + xscale*LngTab(j)
         pyLast = yshift + yscale*LatTab(j)

c.......compute the window coordinates of the next data point

```

```
         pxNext = xshift + xscale*LngTab(k)
         pyNext = yshift + yscale*LatTab(k)

c........compute the window coordinates associated with the tic mark

         pxRef(i) = xshift + xscale*xLng
         pyRef(i) = yshift + yscale*yLat

c........calculate a unit vector perpendicular to the flight path curve at the
c        tic mark

         ux     = pyLast - pyNext
         uy     = pxNext - pxLast
         rmag   = sqrt ( ux*ux + uy*uy )
         if ( rmag.gt.0.0 ) then
            uxRef(i) = ux/rmag
            uyRef(i) = uy/rmag
         end if

c........draw the tic mark from one side of the curve to the other

         iPxRef = inint(pxRef(i))
         iPyRef = inint(pyRef(i))

         if ( iPxRef.ge.iPxMin .and. iPxRef.le.iPxMax .and.
     .        iPyRef.ge.iPyMin .and. iPyRef.le.iPyMax ) then

            pxout  = pxRef(i) - 0.5*uxRef(i)*float(lticmj)
            pyout  = pyRef(i) - 0.5*uyRef(i)*float(lticmj)
            call MovAbs ( inint(pxout) , inint(pyout) )

            pxout  = pxRef(i) + 0.5*uxRef(i)*float(lticmj)
            pyout  = pyRef(i) + 0.5*uyRef(i)*float(lticmj)
            call DrwAbs ( inint(pxout) , inint(pyout) )

         end if

      end do

      call picComment ( %val(picGroupEnd) , %val(0) , %val(nil) )

c.....output and group the graduation values

      call picComment ( %val(picGroupBeg) , %val(0) , %val(nil) )
      do i = 1 , nGrads
         write(ChrDat,1) grVal(i)
         call GetStringWidth ( ChrDat , 255 , nchar , iwidth , ixchar )
         TxtOut = ChrDat(1:nchar)
         pxout  = pxRef(i) - 0.5*uxRef(i)*float(lticmj)
     .            - 0.5*float(iWidth)
         pyout  = pyRef(i) - 0.5*abs(uyRef(i))*float(lticmj)
     .            - float(FontData.ascent+FontData.descent)
         iPxRef = inint(pxRef(i))
         iPyRef = inint(pyRef(i))
         if ( iPxRef.ge.iPxMin .and. iPxRef.le.iPxMax .and.
     .        iPyRef.ge.iPyMin .and. iPyRef.le.iPyMax ) then
            call MovAbs ( inint(pxout) , inint(pyout) )
            call DrawString ( %val(TxtOut) )
         end if
      end do
      call picComment ( %val(picGroupEnd) , %val(0) , %val(nil) )

    1 format ( f6.1 )

```

```
      return
      end


c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main

c-----------------------------------------------------------------------
Segment Main
      subroutine DoAppleMenu ( menuItem )
c-----------------------------------------------------------------------
c
      implicit none
!!SETC USINGINCLUDES = FALSE

c.....include alert definitions, Apple menu definitions

      include 'Alert.inc'
      include 'AppleMenu.inc'

c.....declare the input argument for the routine

      integer*2 menuItem

c.....declare return argument for toolbox "Alert" routine

      integer*2 ItemHit

c.....set up variables for handling DAs

      string*255 daName
      integer*2 daRefNum

c.....set up a structure to save the port while calling DAs

      record / GrafPtr / SavePort

c-----------------------------------------------------------------------

c.....use Language System FORTRAN "select case" extension

      select case (menuItem)

c.......we have selected the About item

        case(AppleItemAboutBDPS)

c.........call Alert toolbox function with nil indicating the default filterproc

          itemHit = Alert( %val(rAboutAlert), %val(nil) )

c.......we must have selected something else (DA, MultiFinder, etc.)

        case default

```

```
          call GetPort( %ref(SavePort) )
          call GetItem( %val(AppleMenuHndl), %val(menuItem), %val(daName) )
          daRefNum = OpenDeskAcc( %val(daName) )
          call SetPort( %ref(SavePort) )

      end select

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main

c--------------------------------------------------------------------------------
Segment Main
      subroutine DoEditMenu ( menuItem )
c--------------------------------------------------------------------------------
c
      implicit none
!!SETC USINGINCLUDES = FALSE

c.....include Edit menu definitions

      include 'EditMenu.inc'

c.....include global definitions

      include 'Globals.inc'

c.....declare the input argument for the routine

      integer*2 menuItem

      record / DialogPtr / dptr

c--------------------------------------------------------------------------------

c.....use Language System FORTRAN "select case" extension

      select case (menuItem)

        case default

      end select

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

```

```
c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main
c--------------------------------------------------------------------------------
Segment Main
      subroutine DoFileMenu ( menuItem )
c--------------------------------------------------------------------------------
c
      implicit none
!!SETC USINGINCLUDES = FALSE

c.....include definitions

      include 'FileInfo.inc'
      include 'FileMenu.inc'
      include 'Globals.inc'
      include 'RunSetup.inc'

c.....declare the input argument for the routine

      integer*2               menuItem

c.....prompt string

      string*255              Prompt

c.....file information parameters

      character*4             FilTyp
      character*4             fMaker
      string*255              FilNam

c.....I/O error flags

      integer*2               ioserr

c--------------------------------------------------------------------------------

c.....use Language System FORTRAN "select case" extension

      select case (menuItem)

        case (FileItemNewMission)

c.........first disable the New and Open Mission buttons - one file at a time
          call MenuSet( FileMenuID, FileItemNewMission,  .false. )
          call MenuSet( FileMenuID, FileItemOpenMission, .false. )
c.........modal dialog for run setup
          call RunSetupDialog

        case (FileItemOpenMission)

c.........call the open file routine
          call SFOpenMissionFile( ioserr, FilTyp, fmaker )
          if (ioserr.eq.1) then
c...........disable the New and Open Mission buttons - one file at a time
            call MenuSet( FileMenuID, FileItemNewMission,  .false. )
            call MenuSet( FileMenuID, FileItemOpenMission, .false. )
c...........set the boolean to indicate that old file is in use
            iGotOldFile = .true.
```

```
c...........modal dialog for run setup
            call RunSetupDialog
c...........set the boolean to indicate that old file is no longer in use
            iGotOldFile = .false.
          end if

        case (FileItemClose)

        case (FileItemQuit)
          call ExitToShell

      end select

      return
      end


c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main

c-----------------------------------------------------------------------
Segment Main
      subroutine DoKeyEvent (theEvent)
c-----------------------------------------------------------------------
c     handle a key-press event

!!SETC USINGINCLUDES = FALSE
      implicit none

c.....common block definition files

      include 'Globals.inc'

c.....character items

      integer*2                   ChCode
      character*1                 Ch

c.....menu pieces (menuResult = merge of menuID and menuItem)

      integer*4 menuResult
      integer*4 itemMask /32767/
      integer*4 menuMask /-16/
      integer*2 menuID
      integer*2 menuItem

      record /EventRecord/        theEvent

c-----------------------------------------------------------------------

c.....get the character

      ChCode = jiand ( theEvent.message , CharCodeMask )

c.....get the character's ASCII representation
```

```

      ch     = char ( ChCode )

c.....check for the Cmd key depression

      ChCode = jiand ( theEvent.modifiers , CmdKey )

      if ( ChCode.ne.0 ) then

        menuResult = MenuKey( %val(ch) )

c.......extract the menu and item numbers from within menuResult

        menuItem = jiand ( menuResult, itemMask )
        menuID   = jishft( menuResult, menuMask )

c.......call my Menu handler with the selection

        if (menuID .ne. 0) call DoMenu (menuID, menuItem)

c.......handle any cut, copy, or paste
        if ( (ch.eq.'x' .or. ch.eq.'X') .and. theInput .ne. nil )
     &    call TECut   ( %ref(theInput) )
        if ( (ch.eq.'c' .or. ch.eq.'C') .and. theInput .ne. nil )
     &    call TECopy  ( %ref(theInput) )
        if ( (ch.eq.'v' .or. ch.eq.'V') .and. theInput .ne. nil )
     &    call TEPaste( %ref(theInput) )

c.....if CmdKey not depressed, then it must be TE

      else if ( theInput .ne. nil ) then

        call TEKey ( %val(ch), %ref(theInput) )

      endif

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-------------------------------------------------------------------------------
      subroutine DoMapMenu ( TheEvent , Option )
c-------------------------------------------------------------------------------

c.....include files

      include 'MapMenu.inc'
      include 'OptFlg.inc'

c.....event record

      record   EventRecord        TheEvent

c.....user option index

      integer*2                   Option
```

```

c.....LongWord containing menu selection parameters

      integer*4                   MenuChoice

c.....LongWord masks

      integer*4                   ItemMask
      integer*4                   MenuMask

c.....character items

      integer*2                   ChCode
      character*1                 Ch

c.....set local integer masks

      data ItemMask            / 32767 /
      data MenuMask            /   -16 /

c--------------------------------------------------------------------------------

c.....handle MouseDown in menuBar

      if ( TheEvent.what.eq.mouseDown ) then
         MenuChoice = MenuSelect ( %val(TheEvent.where) )
      end if

c.....handle keyDown events

      if ( TheEvent.what.eq.keyDown ) then
         chcode = jiand ( TheEvent.message , CharCodeMask )
         ch     = char ( chcode )

c........Enter or Return: Make a new Map

         if ( chcode.eq.3 .or. chcode.eq.13 ) then
            call HiliteMenu ( %val(MapMenuID) )
            menu       = MapMenuID
            item       = itemNewMap
            MenuChoice = item + jishft ( menu , - MenuMask )

c........If command key was depressed, activate equivalent menu item

         else
            chcode = jiand ( TheEvent.modifiers , CmdKey )
            if ( chcode.ne.0 ) then
               MenuChoice = MenuKey ( %val(ch) )

c........Otherwise perform no operation

            else
               MenuChoice = 0
            end if
         end if
      end if

c.....determine which menu and menu item were selected

      if ( MenuChoice.gt.0 ) then
         item   = jiand  ( MenuChoice , ItemMask )
         menu   = jishft ( MenuChoice , MenuMask )

c........Options menu
```

```

         if ( menu.eq.MapMenuID ) then

c...........get a new data set to Map

            if ( item.eq.itemGetNewDataSet ) then
               call ReadFlightData
               Option = oCycle
            end if

c...........resize the map window

            if ( item.eq.itemResizeTheMap ) then
               call ResizeTheMap
               Option = oCycle
            end if

c...........make new Map

            if ( item.eq.itemNewMap ) then
               Option = oNew
            end if

c...........save Map into Pict file and redraw the Map window

            if ( item.eq.itemSaveMap ) then
               call SaveTheMap
               Option = oCycle
            end if

c...........redraw the Map window

            if ( item.eq.itemRedraw ) then
               Option = oRedraw
            end if

c...........quit from the program

            if ( item.eq.itemDone ) then
               Option = oQuit
            end if

         end if

      end if

c.....turn off all menu highlighting

      call HiliteMenu ( 0 )

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

```

```
!!S Main

c-------------------------------------------------------------------------------
Segment Main
      subroutine DoMenu ( menuID, menuItem )
c-------------------------------------------------------------------------------
c
      implicit none
!!SETC USINGINCLUDES = FALSE

c.....common block definition files

      include 'AppleMenu.inc'
      include 'FileMenu.inc'
      include 'EditMenu.inc'
      include 'MapMenu.inc'
      include 'MBar.inc'
      include 'Globals.inc'

c.....menu pieces passed in as arguments

      integer*2 menuID
      integer*2 menuItem

c-------------------------------------------------------------------------------

c.....use Language System FORTRAN "select case" extension

      select case (menuID)

c.......choice from Apple menu

        case(AppleMenuID)

          call DoAppleMenu ( menuItem )

c.......choice from File menu

        case(FileMenuID)

          call DoFileMenu ( menuItem )

c.......choice from Edit menu

        case(EditMenuID)

          call DoEditMenu ( menuItem )

c.......choice from Plot menu

        case(MapMenuID)

          call DoMapMenu ( menuItem )

      end select

c.....turn off high-lighting for the selected menu

      call HiliteMenu( %val(0) )

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main

c-------------------------------------------------------------------------
Segment Main
      subroutine DoPlotMenu ( menuItem )
c---------------------------------  --------------------------------------
c
      implicit none
!!SETC USINGINCLUDES = FALSE

c.....include Plot menu definitions

      include 'PlotMenu.inc'

c.....include global definitions

      include 'Globals.inc'

c.....declare the input argument for the routine

      integer*2 menuItem

       record / DialogPtr / dptr

c-------------------------------------------------------------------------

c.....use Language System FORTRAN "select case" extension

      select case (menuItem)

        case default

      end select

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-------------------------------------------------------------------------
      subroutine DrawFlightPath ( xscale , xshift , yscale , yshift )
c-------------------------------------------------------------------------
c     This subroutine overlays a flight path over the map.
c
c     developer:    David F. Smith
c     date:         February 1991

```

```
      include 'CrvDat.inc'
      include 'FntCom.inc'
      include 'LatCom.inc'
      include 'LngCom.inc'
      include 'MapCom.inc'
      include 'MapMenu.inc'
      include 'MapLim.inc'
      include 'OptFlg.inc'
      include 'PenCom.inc'
      include 'PicGrp.inc'
      include 'PntAbs.inc'
      include 'TicDat.inc'
      include 'TrjCom.inc'
      include 'VuWind.inc'
      include 'WinLim.inc'

c.....Rectangle records

      record / rect /           bounds

c.....Picture record handle and pointer

      common / pict    /        PictHndl
      record / PicHandle /      PictHndl
      record / PicPtr /         PictPtr

c.....set up pointer for QuickDraw globals

      common   / QDGPtr /       QDG
      pointer / QDGlobals /     QDG

c.....set pen size for flight path

      call PenSize ( %val(2) , %val(2) )

c.....group the flight path data points

      call picComment ( %val(picGroupBeg) , %val(0) , %val(nil) )
      do i = 1 , ntrpts

c........1st point

          if ( i.eq.1 ) then
             pxout   = xshift + xscale*LngTab(i)
             pyout   = yshift + yscale*LatTab(i)
             call MovAbs ( inint(pxout) , inint(pyout) )

c........all other points

          else if ( i.ne.1 ) then

c...........a longitude discontinuity occurred in the Westward direction

             if ( JmpTab(i).lt.0 ) then
                pxout   = xshift + xscale*( LngTab(i) - 360.0 )
                pyout   = yshift + yscale*LatTab(i)
                call DshAbs ( inint(pxout) , inint(pyout) , icurve )
                pxout   = xshift + xscale*( LngTab(i-1) + 360.0 )
                pyout   = yshift + yscale*LatTab(i-1)
                call MovAbs ( inint(pxout) , inint(pyout) )
                pxout   = xshift + xscale*LngTab(i)
                pyout   = yshift + yscale*LatTab(i)
                call DshAbs ( inint(pxout) , inint(pyout) , icurve )

```

```
c...........no longitude discontinuity occurred

           else if ( JmpTab(i).eq.0 ) then
              pxout  = xshift + xscale*LngTab(i)
              pyout  = yshift + yscale*LatTab(i)
              call DshAbs ( inint(pxout) , inint(pyout) , icurve )

c...........a longitude discontinuity occurred in the Eastward direction

           else if ( JmpTab(i).gt.0 ) then
              pxout  = xshift + xscale*( LngTab(i) + 360.0 )
              pyout  = yshift + yscale*LatTab(i)
              call DshAbs ( inint(pxout) , inint(pyout) , icurve )
              pxout  = xshift + xscale*( LngTab(i-1) - 360.0 )
              pyout  = yshift + yscale*LatTab(i-1)
              call MovAbs ( inint(pxout) , inint(pyout) )
              pxout  = xshift + xscale*LngTab(i)
              pyout  = yshift + yscale*LatTab(i)
              call DshAbs ( inint(pxout) , inint(pyout) , icurve )
           end if
        end if
     end do
     call picComment ( %val(picGroupEnd) , %val(0) , %val(nil) )

c.....reset pen size

     call PenSize ( %val(1) , %val(1) )

c.....display time tic marks

     if ( TimeTics .and. ntrpts.gt.1 ) then
        call DisplayTimeTics ( xscale , xshift , yscale , yshift )
     end if

     return
     end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

!!S DrawTheMap
c-----------------------------------------------------------------------------
      subroutine DrawTheMap ( Option )
c-----------------------------------------------------------------------------
c     This subroutine draws the global map data in the graphics window and
c     overlays a flight path over it.
c
c     developer:    David F. Smith
c     date:         February 1991

      include 'CrvDat.inc'
      include 'FntCom.inc'
      include 'LatCom.inc'
      include 'LngCom.inc'
      include 'MapCom.inc'
      include 'MapMenu.inc'
      include 'MapLim.inc'
      include 'OptFlg.inc'
      include 'PenCom.inc'
      include 'PicGrp.inc'
      include 'PntAbs.inc'
      include 'TicDat.inc'
      include 'TrjCom.inc'
      include 'VuWind.inc'
      include 'WinLim.inc'

c.....Rectangle records

      record / rect /          bounds

c.....graphics window records

      common / MapWindow /     MapWPtr
      record / WindowPtr /     MapWptr

c.....Map window title

      string*255               MapWTitle

c.....Picture record handle and pointer

      common / pict   /        PictHndl
      record / PicHandle /     PictHndl
      record / PicPtr /        PictPtr

c.....set up pointer for QuickDraw globals

      common  / QDGPtr /       QDG
      pointer / QDGlobals /    QDG
      integer*4                jQDGlobals
      external                 jQDGlobals

c.....Intermediate text string

      string*255               TxtOut

c.....character strings

      character*255            ChrDat
      character*9              cscale
      character*9              cshift
      character*1              onechr

c.....dialog interface variables ( note that pointers are i*4 )

      integer*4                nil

c.....define logical*1 items to emulate Boolean Pascal items

      logical*1                visibl
      logical*1                goAway
      logical*1                fUpdate

c.....temporary table of x window coordinates for character placement

      integer*2                ixchar(255)

c.....user option flag

      integer*2                Option

c.....calculate graph window limits
```

```

      iGxMax = iGxMin + inint ( MapHRes*MapWidth  )
      iGyMax = iGyMin + inint ( MapVRes*MapHeight )

c.....set draw window limits

      iWxMin = iGxMin + 35
      iWxMax = iGxMax - 15
      iWyMin = iGyMin + 20
      iWyMax = iGyMax - 24

c.....create a visible graphics window with no goAway box and an invisible
c     analog for refreshing

      if ( iGotMapWptr.eq.0 ) then
         bounds.left   = iGxMin
         bounds.right  = iGxMax
         bounds.top    = iGyMin + 38
         bounds.bottom = iGyMax + 38
         visibl        = .true.
         goAway        = .false.
         MapWTitle    = 'Map Window'
         MapWptr       = NewWindow ( %val(nil) , %val(bounds) ,
     .                               %val(MapWTitle) , %val(visibl) ,
     .                               %val(noGrowDocProc) , %val(int4(-1)) ,
     .                               %val(goAway) , %val(nil) )
         iGotMapWptr   = 1
      else
         fUpdate = .true.
         mWidth  = iGxMax - iGxMin
         mHeight = iGyMax - iGyMin
         call SizeWindow  ( %val(MapWptr) , %val(mWidth) , %val(mHeight) ,
     .                      %val(fUpdate) )
      end if

c.....incorporate the entire window into the update region

      call InvalRect ( %val(MapWPtr.WP^.portRect) )

c.....initiate Plot window update

      call BeginUpdate ( %val(MapWptr) )

c.....open drawing port

      call SetPort  ( %val(MapWptr) )
      call ShowWindow  ( %val(MapWptr) )

c.....set clip and visible region boundaries to port rectangle

      MapWPtr.WP^.clipRgn.RgnH^.RgnP^.RgnBBox = MapWPtr.WP^.portRect
      MapWPtr.WP^.visRgn.RgnH^.RgnP^.RgnBBox  = MapWPtr.WP^.portRect

c.....erase the visible region

      call EraseRect ( %val(MapWPtr.WP^.portRect) )

c.....Open picture in case user wishes to save Map

      PictHndl = OpenPicture ( %val(MapWPtr.WP^.portRect) )
      call ShowPen

c.....set font size to Geneva 9 and get font information

```

```
      FntNam = 'Geneva'
      call GetFNum ( %val(FntNam) , FntNum )
      call TextFont ( %val(FntNum) )
      call TextSize ( %val(9) )
      call GetFontInfo ( %ref(FontData) )

c.....Set initial pointer to bottom left corner of Map window

      ixabs  = iWxMin
      iyabs  = iWyMin

c.....set pen size for drawing map

      call PenSize ( %val(1) , %val(1) )

c.....set Map window screen limits

      ipxmin = iWxMin
      ipxmax = iWxMax
      ipymin = iWyMin
      ipymax = iWyMax

c.....determine spans of Map data

      xmSpan  = xMapMx - xMapMn
      ymSpan  = yMapMx - yMapMn

c.....determine number of major and minor tic marks

      xTicMj = inint ( xmSpan/xDivMj )
      xTicMi = inint ( xDivMj/xDivMi )
      yTicMj = inint ( ymSpan/yDivMj )
      yTicMi = inint ( yDivMj/yDivMi )

c.....determine scale factors which will scale the data to fill the
c     Map window

      xscale = float ( ipxmax - ipxmin )/xmSpan
      yscale = float ( ipymax - ipymin )/ymSpan

c.....determine offsets which will shift the scaled data into the Map
c     window

      xshift = float(ipxmin) - xscale*xMapMn
      yshift = float(ipymin) - yscale*yMapMn

c.....loop on data points to draw curve and make sure it is grouped

      call picComment ( %val(picGroupBeg) , %val(0) , %val(nil) )

      do i = 1 , 13120

c........determine Map window coordinates of data point

         pxout  = xshift + xscale*Longitude(i)
         pyout  = yshift + yscale*Latitude(i)

c........either draw a line segment or position the Map pointer

         if ( PenCommand(i).eq.0 ) then
            call MovAbs ( inint(pxout) , inint(pyout) )
         else if ( PenCommand(i).eq.1 ) then
            call DshAbs ( inint(pxout) , inint(pyout) , icurve )
         end if

      end do
      call picComment ( %val(picGroupEnd) , %val(0) , %val(nil) )

c.....form the Map window frame

      call picComment ( %val(picGroupBeg) , %val(0) , %val(nil) )
      if ( iMapyp.eq.1 .or. icurve.eq.ncurve ) then
         call MovAbs ( ipxmin , ipymin )
         call DrwAbs ( ipxmax , ipymin )
         call DrwAbs ( ipxmax , ipymax )
         call DrwAbs ( ipxmin , ipymax )
         call DrwAbs ( ipxmin , ipymin )
      end if
      call picComment ( %val(picGroupEnd) , %val(0) , %val(nil) )

c.....draw x axis major tic marks or vertical grid lines and calculate
c     associated reference values

      call picComment ( %val(picGroupBeg) , %val(0) , %val(nil) )
      do i = 1 , xTicMj + 1
         xrefmj(i) = xMapMn + xDivMj*float(i-1)
         hticmj(i) = inint ( xshift + xscale*xrefmj(i) )
         if ( GridLines.eq.1 ) then
            if ( i.gt.1 .and. i.le.xTicMj ) then
            call MovAbs ( hticmj(i) , ipymin )
            call DrwAbs ( hticmj(i) , ipymax )
            end if
         else
            call MovAbs ( hticmj(i) , ipymin          )
            call DrwAbs ( hticmj(i) , ipymin + lticmj )
            call MovAbs ( hticmj(i) , ipymax - lticmj )
            call DrwAbs ( hticmj(i) , ipymax          )
         end if
      end do
      call picComment ( %val(picGroupEnd) , %val(0) , %val(nil) )

c.....draw y axis major tic marks or horizontal grid lines and calculate
c     associated reference values

      call picComment ( %val(picGroupBeg) , %val(0) , %val(nil) )
      do i = 1 , yTicMj + 1
         yrefmj(i) = yMapMn + yDivMj*float(i-1)
         vticmj(i) = inint ( yshift + yscale*yrefmj(i) )
         if ( GridLines.eq.1 ) then
            if ( i.gt.1 .and. i.le.yTicMj ) then
            call MovAbs ( ipxmin , vticmj(i) )
            call DrwAbs ( ipxmax , vticmj(i) )
            end if
         else
            call MovAbs ( ipxmin          , vticmj(i) )
            call DrwAbs ( ipxmin + lticmj , vticmj(i) )
            call MovAbs ( ipxmax - lticmj , vticmj(i) )
            call DrwAbs ( ipxmax          , vticmj(i) )
         end if
      end do
      call picComment ( %val(picGroupEnd) , %val(0) , %val(nil) )

c.....draw each of the x axis minor tic marks

      call picComment ( %val(picGroupBeg) , %val(0) , %val(nil) )
      do i = 1 , xTicMj
         do j = 1 , xTicMi - 1
            xrefmi    = xrefmj(i) + xDivMi*float(j)
            hticmi(i,j) = inint ( xshift + xscale*xrefmi )
            call MovAbs ( hticmi(i,j) , ipymin          )
            call DrwAbs ( hticmi(i,j) , ipymin + lticmi )
            call MovAbs ( hticmi(i,j) , ipymax - lticmi )
            call DrwAbs ( hticmi(i,j) , ipymax          )
         end do
      end do
      call picComment ( %val(picGroupEnd) , %val(0) , %val(nil) )

c.....draw each of the y axis minor tic marks

      call picComment ( %val(picGroupBeg) , %val(0) , %val(nil) )
      do i = 1 , yTicMj
         do j = 1 , yTicMi - 1
            yrefmi      = yrefmj(i) + yDivMi*float(j)
            vticmi(i,j) = inint ( yshift + yscale*yrefmi )
            call MovAbs ( ipxmin          , vticmi(i,j) )
            call DrwAbs ( ipxmin + lticmi , vticmi(i,j) )
            call MovAbs ( ipxmax - lticmi , vticmi(i,j) )
            call DrwAbs ( ipxmax          , vticmi(i,j) )
         end do
      end do
      call picComment ( %val(picGroupEnd) , %val(0) , %val(nil) )

c.....output reference values associated with x axis major tic marks

      call picComment ( %val(picGroupBeg) , %val(0) , %val(nil) )
      do ix = 1 , xTicMj + 1
         write(ChrDat,1) xrefmj(ix)/10.0**ilogx
         call GetStringWidth ( ChrDat , 255 , nchar , iwidth , ixchar )
         ipxref = hticmj(ix) - iwidth/2
         ipyref = ipymin - 3*FontData.ascent/2
         call MovAbs ( ipxref , ipyref )
         TxtOut = ChrDat(1:nchar)
         call DrawString ( %val(TxtOut) )
      end do
      call picComment ( %val(picGroupEnd) , %val(0) , %val(nil) )

c.....output reference values associated with y axis major tic marks

      call picComment ( %val(picGroupBeg) , %val(0) , %val(nil) )
      do iy = 1 , yTicMj + 1
         call MovAbs ( ipxmin , vticmj(iy) )
         write(ChrDat,1) yrefmj(iy)/10.0**ilogy
         call GetStringWidth ( ChrDat , 255 , nchar , iwidth , ixchar )
         call MovRel (   iwidth , - FontData.ascent/2 )
         TxtOut = ChrDat(1:nchar)
         call DrawString ( %val(TxtOut) )
      end do
      call picComment ( %val(picGroupEnd) , %val(0) , %val(nil) )

c.....draw the flight path as well

      if ( ntrpts.ge.1 ) then
         call DrawFlightPath ( xscale , xshift , yscale , yshift )
      end if

c.....close the picture frame and unlock the picture handle

      call [illegible]
      call ClosePicture
      call HUnlock ( %val(picHndl) )

c.....terminate the update process
```

```

      call EndUpdate ( %val(MapWPtr) )

    1 format ( f6.1 )

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-----------------------------------------------------------------------
      subroutine DrwAbs ( ix , iy )
c-----------------------------------------------------------------------
c     Draw to absolute graphics window position .

      include 'PntAbs.inc'
      include 'WinLim.inc'

      ixabs  = ix
      iyabs  = iy

      call LineTo ( %val(ixabs) , %val(iGyMax-iyabs) )

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-----------------------------------------------------------------------
      subroutine DshAbs ( ix , iy , icurve )
c-----------------------------------------------------------------------
c     Draw a dashed line to the point (ipx,ipy) from current pen location,
c     clipping the portion outside the Map window.

      include 'CrvDat.inc'
      include 'PntAbs.inc'
      include 'WinLim.inc'

c.....return if pen location will not change

      if ( ix.eq.ixabs .and. iy.eq.iyabs ) then
         return
      end if

c.....initialize the end points of the segment to draw

      ix(1)   = ixabs
      iy(1)   = iyabs
      ixnew   = ix
      iynew  = iy

      xold   = float ( ixold )
      yold   = float ( iyold )
      xnew   = float ( ixnew )
      ynew   = float ( iynew )

c.....see if old pen location is inside the Map window

      if ( ipxmin.le.ixold .and. ipxmax.ge.ixold .and.
     .     ipymin.le.iyold .and. ipymax.ge.iyold ) then
         locold = 0
      else
         locold = 1
      end if

c.....see if current pen location is inside the Map window

      if ( ipxmin.le.ixnew .and. ipxmax.ge.ixnew .and.
     .     ipymin.le.iynew .and. ipymax.ge.iynew ) then
         locnew = 0
      else
         locnew = 1
      end if

c.....current pen location is inside window, new pen location is outside window.
c     determine where line segment intersects window boundary.

      if ( locold.eq.0 .and. locnew.eq.1 ) then

c........horizontal line intersects either left or right boundary of window

         if ( iyold.eq.iynew ) then
            locnew = 0
            ixnew  = max0 ( ipxmin , min0 ( ipxmax , ixnew ) )
         end if

c........vertical line intersects either lower or upper boundary of window

         if ( ixold.eq.ixnew ) then
            locnew = 0
            iynew  = max0 ( ipymin , min0 ( ipymax , iynew ) )
         end if

c........sloped line can intersect any of the four boundaries

         if ( ixold.ne.ixnew .and. iyold.ne.iynew ) then

c...........determine slope and x intercept

            slope  = ( ynew - yold )/( xnew - xold )
            b      = ynew - xnew*slope

c...........see if left boundary is intersected

            if ( ixnew.lt.ipxmin ) then
               iyhit  = inint ( b + slope*float(ipxmin) )
               if ( iyhit.ge.ipymin .and. iyhit.le.ipymax ) then
                  locnew = 0
                  ixnew  = ipxmin
                  iynew  = iyhit
               end if
            end if
```

```
c...........see if right boundary is intersected

            if ( ixnew.gt.ipxmax ) then
               iyhit  = inint ( b + slope*float(ipxmax) )
               if ( iyhit.ge.ipymin .and. iyhit.le.ipymax ) then
                  locnew = 0
                  ixnew  = ipxmax
                  iynew  = iyhit
               end if
            end if

c...........see if lower boundary is intersected

            if ( iynew.lt.ipymin ) then
               ixhit  = inint ( ( float(ipymin) - b )/slope )
               if ( ixhit.ge.ipxmin .and. ixhit.le.ipxmax ) then
                  locnew = 0
                  ixnew  = ixhit
                  iynew  = ipymin
               end if
            end if

c...........see if upper boundary is intersected

            if ( iynew.gt.ipymax ) then
               ixhit  = inint ( ( float(ipymax) - b )/slope )
               if ( ixhit.ge.ipxmin .and. ixhit.le.ipxmax ) then
                  locnew = 0
                  ixnew  = ixhit
                  iynew  = ipymax
               end if
            end if

         end if
      end if

c.....current pen location is outside window, new pen location is inside window.
c     determine where line segment intersects window boundary.

      if ( locold.eq.1 .and. locnew.eq.0 ) then

c........horizontal line intersects either left or right boundary of window

         if ( iyold.eq.iynew ) then
            locold = 0
            ixold  = max0 ( ipxmin , min0 ( ipxmax , ixold ) )
         end if

c........vertical line intersects either lower or upper boundary of window

         if ( ixold.eq.ixnew ) then
            locold = 0
            iyold  = max0 ( ipymin , min0 ( ipymax , iyold ) )
         end if

c........sloped line can intersect any of the four boundaries

         if ( ixold.ne.ixnew .and. iyold.ne.iynew ) then

c...........determine slope and x intercept

            slope  = ( ynew - yold ) / ( xnew - xold )
            b      = ynew - xnew*slope
```

```
c...........see if left boundary is intersected

            if ( ixold.lt.ipxmin ) then
               iyhit  = inint ( b + slope*float(ipxmin) )
               if ( iyhit.ge.ipymin .and. iyhit.le.ipymax ) then
                  locold = 0
                  ixold  = ipxmin
                  iyold  = iyhit
               end if
            end if

c...........see if right boundary is intersected

            if ( ixold.gt.ipxmax ) then
               iyhit  = inint ( b + slope*float(ipxmax) )
               if ( iyhit.ge.ipymin .and. iyhit.le.ipymax ) then
                  locold = 0
                  ixold  = ipxmax
                  iyold  = iyhit
               end if
            end if

c...........see if lower boundary is intersected

            if ( iyold.lt.ipymin ) then
               ixhit  = inint ( ( float(ipymin) - b )/slope )
               if ( ixhit.ge.ipxmin .and. ixhit.le.ipxmax ) then
                  locold = 0
                  ixold  = ixhit
                  iyold  = ipymin
               end if
            end if

c...........see if upper boundary is intersected

            if ( iyold.gt.ipymax ) then
               ixhit  = inint ( ( float(ipymax) - b )/slope )
               if ( ixhit.ge.ipxmin .and. ixhit.ge.ipxmax ) then
                  locold = 0
                  ixold  = ixhit
                  iyold  = ipymax
               end if
            end if

         end if
      end if

c.....both end points are outside the window.  Determine whether line segment
c     intersects window boundaries.

      if ( locold.eq.1 .and. locnew.eq.1 ) then

c........see if horizontal line intersects either left or right boundary of
c        window

         if ( iyold.eq.iynew ) then
            if ( iyold.gt.ipymin .and. iyold.lt.ipymax ) then
               locold = 0
               locnew = 0
               if ( ixold.le.ipxmin ) then
                  ixold  = ipxmin
                  ixnew  = ipxmax
               else if ( ixold.ge.ipxmax ) then
                  ixold  = ipxmax
                  ixnew  = ipxmin
               end if
            end if
         end if

c........see if vertical line intersects either lower or upper boundary of
c        window

         if ( ixold.eq.ixnew ) then
            if ( ixold.gt.ipxmin  .and. ixold.lt.ipxmax ) then
               locold = 0
               locnew = 0
               if ( iyold.le.ipymin ) then
                  iyold  = ipymin
                  iynew  = ipymax
               else if ( iyold.ge.ipymax ) then
                  iyold  = ipymax
                  iynew  = ipymin
               end if
            end if
         end if

c........sloped line can intersect any of the four boundaries

         if ( ixold.ne.ixnew .and. iyold.ne.iynew ) then

c...........determine slope and x intercept

            slope  = ( ynew - yold )/( xnew - xold )
            b      = ynew - xnew*slope

c...........see if left boundary is intersected

            iyhit  = inint ( b + slope*float(ipxmin) )
            if ( iyhit.ge.ipymin .and. iyhit.le.ipymax ) then
               if ( ixold.lt.ixnew ) then
                  locold = 0
                  ixold  = ipxmin
                  iyold  = iyhit
               else if ( ixold.gt.ixnew ) then
                  locnew = 0
                  ixnew  = ipxmin
                  iynew  = iyhit
               end if
            end if

c...........see if right boundary is intersected

            iyhit  = inint ( b + slope*float(ipxmax) )
            if ( iyhit.ge.ipymin .and. iyhit.le.ipymax ) then
               if ( ixold.lt.ixnew ) then
                  locnew = 0
                  ixnew  = ipxmax
                  iynew  = iyhit
               else if ( ixold.gt.ixnew ) then
                  locold = 0
                  ixold  = ipxmax
                  iyold  = iyhit
               end if
            end if

c...........see if lower boundary is intersected

            ixhit  = inint ( ( float(ipymin) - b )/slope )
            if ( ixhit.ge.ipxmin .and. ixhit.le.ipxmax ) then
               if ( iyold.lt.iynew ) then
                  locold = 0
                  ixold  = ixhit
                  iyold  = ipymin
               else if ( iyold.gt.iynew ) then
                  locnew = 0
                  ixnew  = ixhit
                  iynew  = ipymin
               end if
            end if

c...........see if upper boundary is intersected

            ixhit  = inint ( ( float(ipymax) - b )/slope )
            if ( ixhit.ge.ipxmin .and. ixhit.le.ipxmax ) then
               if ( iyold.lt.iynew ) then
                  locnew = 0
                  ixnew  = ixhit
                  iynew  = ipymax
               else if ( iyold.gt.iynew ) then
                  locold = 0
                  ixold  = ixhit
                  iyold  = ipymax
               end if
            end if

         end if
      end if

c.....set the rectangle limits containing the line segment

      ilxmin = min0 ( ix , ixabs )
      ilxmax = max0 ( ix , ixabs )
      ilymin = min0 ( iy , iyabs )
      ilymax = max0 ( iy , iyabs )

c.....draw visible part of dashed line segment

      if ( locold.eq.0 .and. locnew.eq.0 ) then

         if ( ixold.ge.ilxmin .and. ixold.le.ilxmax .and.
     .        iyold.ge.ilymin .and. iyold.le.ilymax .ard.
     .        ixnew.ge.ilxmin .and. ixnew.le.ilxmax .and.
     .        iynew.ge.ilymin .and. iynew.le.ilymax ) then

            call DashIt ( ixold , iyold , ixnew , iynew , icurve )

         end if

      end if

c.....save the current unclipped pen location

      ixabs  = ix
      iyabs  = iy

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc
```

```

c.....Load the ToolBox traps

!!M Inlines.f

c---------------------------------------------------------------------------------
      subroutine EventHandler ( TheEvent , Option )
c---------------------------------------------------------------------------------
c     This routines figures out what kind of event has occurred and
c     calls the appropriate routine to take action in response to the event.

c.....user option flags

      include 'OptFlg.inc'

      record / EventRecord /     TheEvent
      record / WindowPtr /       Wptr

      integer*2                  Option
      integer*2                  WindowPart

c---------------------------------------------------------------------------------

c.....set option flag to continue

      Option = oCycle

c.....event is MouseDown in menu region

      if ( TheEvent.what.eq.mouseDown ) then
         WindowPart = FindWindow ( %val(TheEvent.where) , Wptr )
         if ( WindowPart.eq.inMenuBar ) then
            call DoMapMenu ( TheEvent , Option )
         end if

c.....key press events

      else if ( TheEvent.what.eq.keyDown ) then
         call DoMapMenu ( TheEvent , Option )

c.....ignore all other events

      else
      end if

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main

c--------------------------------------------------------------------------------
Segment Main
      subroutine eventloop
```

```
c-----------------------------------------------------------------------
c     get events forever and handle them.

!!SETC USINGINCLUDES = FALSE
      implicit none

c.....common block definition files

      include 'AppleMenu.inc'
      include 'FileMenu.inc'
      include 'EditMenu.inc'
      include 'MapMenu.inc'
      include 'MBar.inc'
      include 'Globals.inc'

c.....define 2-byte zero for argument to FlushEvents toolbox routine

      integer*2 zero /0/

c.....character items

      integer*2                 ChCode
      character*1               Ch

c.....menu pieces (menuResult = merge of menuID and menuItem)

      integer*4 menuResult
      integer*4 itemMask /32767/
      integer*4 menuMask /-16/
      integer*2 menuID
      integer*2 menuItem

c.....code for event type

      integer*2 code

c.....boolean for getting events

      logical*1 gotEvent

c.....structures for window, event

      record / WindowPtr   /      whichWindow
      record / EventRecord /      theEvent

c-----------------------------------------------------------------------

c.....we will start fresh on events

      call FlushEvents ( %val(everyEvent), %val(zero) )

c.....enable the menus for "New Mission", "Open Mission"

      call MenuSet ( FileMenuID, FileItemNewMission,  .true. )
      call MenuSet ( FileMenuID, FileItemOpenMission, .true. )

c.....set the flag for exiting program; we're not done yet

      doneFlag = .false.

c.....initialize to no active text edit selection

      theInput.TEH = nil

```

```
c.....***********************
c.....*** Main Event Loop ***
c.....***********************

      do while (.true.)

c.....see if a TE is active

      if ( theInput.TEH .ne. nil ) call TEIdle( %ref(theInput) )

c.....get a MultiFinder event

      if (gHasWaitNextEvent) then
        gotEvent = WaitNextEvent( %val(everyEvent), %ref(theEvent),
     &                            int4(SleepValue), %val(nil)     )
      else
        call SystemTask
        gotEvent = GetNextEvent( %val(everyEvent), %ref(theEvent))
      end if

c.....work the event

      if (gotEvent) then
        select case (theEvent.what)

          case (mouseDown)

c...........find the window in which the event happened
            code = FindWindow( %val(theEvent.where), %ref(whichWindow) )

            select case (code)
              case (inMenuBar)

c...............determine the menu selection
                menuResult = MenuSelect ( %ref(theEvent.where) )

c...............extract the menu and item numbers from within menuResult
                menuItem = jiand ( menuResult, itemMask )
                menuID   = jishft( menuResult, menuMask )
c...............call my Menu handler
                call DoMenu (menuID, menuItem)

              case (inDrag)
cc              call DoDrag(whichWindow)

              case (inGrow)
cc              call DoGrow(whichWindow)

              case (inGoAway)
cc              call DoGoAway(whichWindow)

              case (inContent)
cc              call DoInContent(whichWindow)

              case (inSysWindow)
                call SystemClick( %ref(theEvent), %ref(whichWindow) )

            end select !code

          case (keyDown, autoKey)
            call DoKeyEvent (theEvent)

          case (UpdateEvt)
cc          call DoUpdate
```

```

              case (DiskEvt)
cc              call DoDiskEvent

              case (ActivateEvt)
cc              call DoActivate

              case default

            end select !theEvent.what

c.....end of if construct for gotEvent

          endif

c.....******************************
c.....*** end of Main Event Loop ***
c.....******************************

          end do

          return
          end
```

```
!!T72+
      SUBROUTINE GEN4D                                                  GEN4   1
C.....GENERATES NG = 9 OR 16 4D PROFILES P,D,T AND SIGMAS SP,SD,ST AT   GEN4   2
C        GRID OF LATITUDES AND LONGITUDES GLAT,GLON. CURRENT LATITUDE,  GEN4   3
C        LONGITUDE=CLAT,CLON. PREVIOUS LATITUDE,LONGITUDE=PLAT,PLON.    GEN4   4
      COMMON/C4/GLAT(16),GLON(16),NG,P(16,26),D(16,26),T(16,26),        GEN4   5
     $ SP(16,26),SD(16,26),ST(16,26),PLON,CLON,HS                       GEN4   6
      COMMON/IOTEMP/IOTEM1,IOTEM2 IUG,IUN,DDD,XMJD,PLAT,CLAT,           GEN4   7
     $ NSAME,RP1,RD1,RT1,SP1,SD1,ST1,RU1,RV1,SU1,SV1,                   GEN4   8
     $ MN,IDA,IYR,H1,PHI1R,THET1R,GZ,RI,Z,PHIR,THETR,F10,F10B,AP,       GEN4   9
     $ IHR,MIN,NMORE,DX,HL,VL,DZ,B,EPS,IOPP,LOOK,DUMMY(20)              GEN4  10
      COMMON/PDTCOM/IU4,MONTH,IOPR,PG(18,19),TG(18,19),DG(18,19),       GEN4  11
     1     PSP(8,10,12),DSP(8,10,12),TSP(8,10,12)                       GEN4  12
     2,PAQ(17,5),DAQ(17,5),TAQ(17,5),                                   GEN4  13
     3PDQ(17,5),DDQ(17,5),TDQ(17,5),PR(20,10),DR(20,10),TR(20,10),      GEN4  14
     4UAQ(17,5),VAQ(17,5),UDQ(17,5),VDQ(17,5),UR(25,10),VR(25,10),      GEN4  15
     5PQ,DQ,TQ,UQ,VQ,PQA,DQA,TQA,UA,VA,IOPQ                             GEN4  16
     * ,PLP(25,10),DLP(25,10),TLP(25,10),ULP(25,10),VLP(25,10),UDL(25,  GEN4  17
     * 10),VDL(25,10),UDS(25,10),VDS(25,10)                             GEN4  18
      COMMON/ADJCOM/DUM(130),KOUNT                                      GEN4  19
      COMMON/IPRTP/ IPRT                                                GEN4  20
      DIMENSION NGOOD(26)                                               GEN4 20B
      IF(NSAME.EQ.1) RETURN                                             GEN4  21
      IPRT=0                                                            GEN4  22
      LOOK=0                                                            GEN4  23
      F = 0.017453293                                                   GEN4  24
      NG = 16                                                           GEN4  25
      DX = PLON - CLON                                                  GEN4  26
      IF(DX.GT.180.0) DX = DX - 360.0                                   GEN4 26B
      IF(DX.LT.-180.0)DX = DX + 360.0                                   GEN4 26C
C.....LONGITUDE DISPLACEMENT FROM PREVIOUS TO CURRENT POSITION          GEN4  27
      DY = CLAT - PLAT                                                  GEN4  28
C.....LATITUDE DISPLACEMENT FROM PREVIOUS TO CURRENT POSITION           GEN4  29
      IF (DY) 20,10,20                                                  GEN4  30
   10 IF (DX) 15,12,15                                                  GEN4  31
   12 K = 0                                                             GEN4  32
      GO TO 40                                                          GEN4  33
   15 THETA = 180. + SIGN(90.,DX)                                       GEN4  34
      GO TO 30                                                          GEN4  35
   20 THETA = ATAN(DX/DY)/F                                           GEN4  36
      IF (DY.GT.0.) THETA = THETA + 180.                              GEN4  37
      IF (THETA.LT.0.) THETA = THETA + 360.                           GEN4  38
C.....THETA = AZIMUTH ANGLE OF TRAJECTORY, USED TO ORIENT LAT-LON GRID GEN4  39
   30 K = INT((THETA + 67.5)/45.)                                     GEN4  40
C     INDEX USED IN COMPUTED GO TO FOR 110 THRU 180                   GEN4  41
      IF (K.GT.8) K=K-8                                               GEN4  42
C     NORTH POLAR GRID                                                GEN4  43
      IF (CLAT.GT.75.0.AND.K.GE.3.AND.K.LE.7)GO TO 200                GEN4  44
C     SOUTH POLAR GRID                                                GEN4  45
      IF (CLAT.LT.-75.0.AND.(K.GE.7.OR.K.LE.3))GO TO 200              GEN4  46
C.....INITIAL ESTIMATE OF REFERNCE LATITUDE (LOWER LEFT GRID POINT)   GEN4  47
 40   LAT0 = 5*INT(CLAT/5.)                                           GEN4  48
      IF (CLAT.LT.0.) LAT0 = LAT0 - 5                                 GEN4  49
C.....INITIAL ESTIMATE OF REFERENCE LONGITUDE (LOWER LEFT GRID POINT)  GEN4  50
      LON0=5*INT(CLON/5.)                                             GEN4  51
C.....ADJUSTS LAT0,LON0 ACCORDING TO DIRECTION OF TRAJECTORY AZIMUTH   GEN4  52
      IF (K.GT.0) GO TO 100                                           GEN4  53
      LAT0 = LAT0 - 5                                                 GEN4  54
      LON0= LON0 + 10                                                 GEN4  55
      GO TO 190                                                       GEN4  56
  100 GO TO (110,120,130,140,150,160,170,180),K                       GEN4  57
  110 LAT0 = LAT0-10                                                  GEN4  58
      LON0 = LON0 + 10                                                GEN4  59
      GO TO 190                                                       GEN4  60
  120 LAT0 = LAT0-10                                                  GEN4  61
      LON0 = LON0+15                                                  GEN4  62
      GO TO 190                                                       GEN4  63
  130 LAT0 = LAT0-5                                                   GEN4  64
      LON0 = LON0+15                                                  GEN4  65
      GO TO 190                                                       GEN4  66
  140 LON0 = LON0+15                                                  GEN4  67
      GO TO 190                                                       GEN4  68
  150 LON0 = LON0+10                                                  GEN4  69
      GO TO 190                                                       GEN4  70
  160 LON0 = LON0+5                                                   GEN4  71
      GO TO 190                                                       GEN4  72
  170 LAT0 = LAT0-5                                                   GEN4  73
      LON0 = LON0+5                                                   GEN4  74
      GO TO 190                                                       GEN4  75
  180 LAT0 = LAT0-10                                                  GEN4  76
      LON0 = LON0+5                                                   GEN4  77
  190 IF (LON0.GE.360) LON0 = LON0 - 360                              GEN4  78
      IF (LAT0.GT.75)  LAT0 = 75                                      GEN4 78B
      DLI=1.25                                                        GEN4  79
      IF(ABS(CLAT).GE.18) GO TO 192                                   GEN4  80
      DLI=3.0                                                         GEN4  81
      LAT0=-18                                                        GEN4  82
  192 DO 195 I=1,4                                                    GEN4  83
      I12 = I+12                                                      GEN4  84
      DO 195 J=I,I12,4                                                GEN4  85
      GLAT(J) = LAT0 + DLI*(J-I)                                      GEN4  86
C.....LATITUDE, LONGITUDE GRID AT 5 DEGREE INTERVALS                  GEN4  87
  195 GLON(J) = LON0 - 5. * (I - 1)                                   GEN4  88
      GO TO 400                                                       GEN4  89
C     POLAR GRID                                                      GEN4  90
  200 NG = 9                                                          GEN4  91
      DO 210 J=1,8                                                    GEN4  92
C.....POLAR GRID LATITUDES 1-8 = +75 (N) OR -75 (N)                   GEN4  93
      GLAT(J) = SIGN(75.,CLAT)                                        GEN4  94
C.....POLAR GRID LONGITUDES 1-8 AT 45 DEG INTERVALS                   GEN4  95
  210 GLON(J) = 45.*(J-1)                                             GEN4  96
C.....POLAR GRID LATITUDE 9 = POLE +93 OR -90                         GEN4  97
      GLAT(9) = SIGN(90.,CLAT)                                        GEN4  98
```

```
C.....POLAR GRID LONGITUDE 9 = 0                                        GEN4  99
      GLON(9) = 0.                                                      GEN4 100
C.....GENERATES 16 PROFILES (OR 9 PROFILES FOR POLAR GRID)              GEN4 101
  400 CALL GRID4D                                                       GEN4 102
      DO 390 I = 1,NG                                                   GEN4102B
      DO 330 J = 1,26                                                   GEN4102C
      NGOOD(J) = 1                                                      GEN4102D
      IF(P(I,J).LE.0.0.OR.D(I,J).LE.0.0.OR.T(I,J).LE.0.0)NGOOD(J)=0     GEN4102E
      IF(NGOOD(J).EQ.0)GOTO 330                                         GEN4102F
      RATIO = P(I,J)/(D(I,J)*T(I,J))                                    GEN4102G
      IF(RATIO.GT.286.0.AND.RATIO.LT.288.0)GOTO 330                     GEN4102H
      NGOOD(J) = 0                                                      GEN4102I
      WRITE(6,325)I,J,RATIO                                             GEN4102J
  325 FORMAT(' GAS LAW VIOLATION. I,J,RATIO = ',2I4,G12.4)              GEN4102K
  330 CONTINUE                                                          GEN4102L
      DO 340 J = 3,26                                                   GEN4102M
      IF(NGOOD(J).EQ.0.OR.NGOOD(J-1).EQ.0)GOTO 340                      GEN4102N
      DENOM = 1./T(I,J)                                                 GEN4102O
      IF(ABS(T(I,J)-T(I,J-1)).GT.0.01)DENOM=ALOG(T(I,J-1)/T(I,J))/      GEN4102P
     & (T(I,J-1)-T(I,J))                                                GEN4102Q
      RATIO = ALOG(P(I,J-1)/P(I,J))/DENOM                               GEN4102R
      IF(RATIO.GT.30.7.AND.RATIO.LT.37.6)GOTO 340                       GEN4102S
      NGOOD(J) = 0                                                      GEN4102T
      WRITE(6,335)I,J,RATIO                                             GEN4102U
  335 FORMAT(' HYDROSTATIC VIOLATION. I,J,RATIO = ',2I4,G12.4)          GEN4102V
      IF(J.EQ.26)GO TO 345                                              GEN4102W
      K1 = J + 1                                                        GEN4102X
      DO 336 K = K1,26                                                  GEN4102Y
  336 NGOOD(K) = 0                                                      GEN4102Z
      GO TO 345                                                         GEN4102a
  340 CONTINUE                                                          GEN4102b
  345 NBAD = 0                                                          GEN4102c
      DO 360 J = 1,26                                                   GEN4102d
      IF(NGOOD(J).GT.0)GOTO 360                                         GEN4102e
      NBAD = NBAD + 1                                                   GEN4102f
      P(I,J) = 0.                                                       GEN4102g
      D(I,J) = 0.                                                       GEN4102h
      T(I,J) = 0.                                                       GEN4102i
  360 CONTINUE                                                          GEN4102j
      IF(NBAD.LE.12)GOTO 390                                            GEN4102k
      WRITE(6,380)                                                      GEN4102l
  380 FORMAT(' UNABLE TO GENERATE 4-D GRID. TOO MANY TEST VIOLATIONS')  GEN4102m
      STOP                                                              GEN4102n
  390 CONTINUE                                                          GEN4102o
      DO 600 I=1,NG                                                     GEN4 103
      IHV = 0                                                           GEN4 104
      SPR = 0.0004                                                      GEN4 105
      SDR = 0.0004                                                      GEN4 106
      STR = 0.0004                                                      GEN4 107
      DO 420 J = 8,26                                                   GEN4 108
      CHECK = 1.                                                        GEN4 109
      IF(P(I,J).LE.0.0.OR.SP(I,J).LE.0.0)CHECK = 0.                     GEN4 110
      IF(D(I,J).LE.0.0.OR.SD(I,J).LE.0.0)CHECK = 0.                     GEN4 111
      IF(T(I,J).LE.0.0.OR.ST(I,J).LE.0.0)CHECK = 0.                     GEN4 112
C.... FINDS INDEX IHV OF LAST HEIGHT ABOVE 6 KM WITH NON-ZERO DATA      GEN4 113
      IF(CHECK.GT.0.)GO TO 420                                          GEN4 114
      IHV = J-1                                                         GEN4 115
      GO TO 440                                                         GEN4 116
  420 CONTINUE                                                          GEN4 117
C     HEIGHT = HEIGHT INDEX - 1                                         GEN4 118
  440 Z1 = IHV -1.                                                      GEN4 119
      IF(IHV.EQ.0)GO TO 491                                             GEN4119B
C     SPR,SDR,STR=SIGMAS AT HEIGHT Z1                                   GEN4 120
      SPR = SP(I,IHV)                                                   GEN4 121
      SDR=SD(I,IHV)                                                     GEN4 122
      STR=ST(I,IHV)                                                     GEN4 123
      IF(SPR.LE.0.0)SPR = 0.0004                                        GEN4123B
      IF(SDR.LE.0.0)SDR = 0.0004                                        GEN4123C
      IF(STR.LE.0.0)STR = 0.0004                                        GEN4123D
      IF(IHV.GT.12)GOTO 441                                             GEN4123E
      WRITE(6,442)IHV                                                   GEN4123F
  442 FORMAT(' UNABLE TO GENERATE 4-D GRID. IHV = ',I3)                 GEN4123G
      STOP                                                              GEN4123H
  441 CONTINUE                                                          GEN4123I
C.....IF HEIGHT Z1 GEQ 20 KM, USE GROVES AT 30 KM FOR INTERPOLATION,     GEN4 124
C         OTHERWISE USE GROVES AT 25 KM                                 GEN4 125
      IF (IHV.GE.21) GO TO 480                                          GEN4 126
C.....EVALUATES GROVES AT 25 KM FOR INTERPOLATION AND                   GEN4 127
C         FILL IN OF ZERO DATA                                          GEN4 128
      CALL GTERP(25,GLAT(I),P2,D2,T2,PG,DG,TG,DPY,DTY,DP2Y)             GEN4 129
      IHP = IHV + 1                                                     GEN4 130
      DO 450 K=IHP,26                                                   GEN4 131
C.....AVOIDS INTERPOLATION OF P,D,T IF ONLY SIGMAS ARE ZERO             GEN4 132
      IF(P(I,K).GT.0.0.AND.D(I,K).GT.0.0.AND.T(I,K).GT.0.0)GO TO 445    GEN4 133
      H=K-1                                                             GEN4 134
C.....INTERPOLATES BETWEEN 4D AT HEIGHT Z1 AND GROVES AT 25 TO FILL     GEN4 135
C         IN MISSING DATA                                               GEN4 136
      CALL INTER2(P(I,IHV),D(I,IHV),T(I,IHV),Z1,P2,D2,T2,25.,PH,DH,TH,H)GEN4 137
      P(I,K)=PH                                                         GEN4 138
      D(I,K)=DH                                                         GEN4 139
      T(I,K)=TH                                                         GEN4 140
  445 SP(I,K) = SPR                                                     GEN4 141
      SD(I,K)=SDR                                                       GEN4 142
C.....SETS MISSING SIGMAS EQUAL TO SIGMAS AT HEIGHT Z1                  GEN4 143
  450 ST(I,K)=STR                                                       GEN4 144
      GO TO 491                                                         GEN4 145
C.....EVALUATES GROVES AT 30 KM FOR INTERPOLATION AND FILL IN OF        GEN4 146
C         ZERO DATA                                                     GEN4 147
  480 CALL GTERP(30,GLAT(I),P2,D2,T2,PG,DG,TG,DPY,DTY,DP2Y)             GEN4 148
C     COMPUTE PERTURBATIONS TO GROVES MODEL                             GEN 148B
      CALL PDTUV(PSP,DSP,TSP,GLAT(I),GLON(I),30,DP,DD,DT,DPX,DPY,DTX,DTYGEN4 149
     $ ,DP2X,DP2Y,DPXY)                                                 GEN4 151
C.....ADD STATIONARY PERTURBATIONS TO GROVES MODEL                      GEN4 152
      P2 = P2*(1. + DP)                                                 GEN4 153
      D2 = D2*(1. + DD)                                                 GEN4 154
      T2 = T2*(1. + DT)                                                 GEN4 155
      IHP = IHV + 1                                                     GEN4 156
      DO 490 K=IHP,26                                                   GEN4 157
C.....AVOIDS INTERPOLATING P,D,T IF ONLY SIGMAS ARE ZERO                GEN4 158
      IF(P(I,K).GT.0.0.AND.D(I,K).GT.0.0.AND.T(I,K).GT.0.0)GO TO 485    GEN4 159
      H=K-1                                                             GEN4 160
C.....INTERPOLATES BETWEEN 4D AT HEIGHT Z1 AND GROVES AT 30 KM TO       GEN4 161
C         FILL IN MISSING DATA                                          GEN4 162
      CALL INTER2(P(I,IHV),D(I,IHV),T(I,IHV),Z1,P2,D2,T2,30.,PH,DH,TH,H)GEN4 163
      P(I,K)=PH                                                         GEN4 164
      D(I,K)=DH                                                         GEN4 165
      T(I,K)=TH                                                         GEN4 166
  485 SP(I,K) = SPR                                                     GEN4 167
      SD(I,K)=SDR                                                       GEN4 168
C     SET MISSING SIGMAS AT HEIGHT 1                                    GEN4 169
  490 ST(I,K) = STR                                                     GEN4 170
  491 CONTINUE                                                          GEN4 171
      IHP = IHV - 1                                                     GEN4 172
      SP0 = SP(I,1)                                                     GEN4 173
      SD0 = SD(I,1)                                                     GEN4 174
      ST0 = ST(I,1)                                                     GEN4 175
      IF(SP0.LE.0.0)SP0 = 0.0001                                        GEN4 176
      IF(SD0.LE.0.0)SD0 = 0.0001                                        GEN4176B
      IF(STO.LE.0.0)STO = 0.0001
      DO 492 K = 1,9
      IF(SP(I,K) .LE. 0.) SP(I,K) = SP0
      IF(SD(I,K) .LE. 0.) SD(I,K) = SD0
  492 IF(ST(I,K) .LE. 0.) ST(I,K) = ST0
      DO 495 K=10,IHP
C......SETS ALL ZERO SIGMAS TO SIGMA AT HEIGHT Z1
      IF (SP(I,K).LE.0.0.AND.P(I,K).GT.0.) SP(I,K) = SPR
      IF (SD(I,K).LE.0.0.AND.D(I,K).GT.0.) SD(I,K) = SDR
  495 IF (ST(I,K).LE.0.0.AND.T(I,K).GT.0.) ST(I,K) = STR
  500 PA = P(I,1)
      TA = T(I,1)
      R =287.05
      G = GZ*(1.+(Z/(RI-Z)))**2
      K = 2
  510 PB = P(I,K)
      TB = T(I,K)
      IF ((PB*TB) .GT. 0.) GO TO 520
      K = K + 1
      GO TO 510
  520 IF(ABS(TA-TB).LE.0.01)GOTO 570
  560 IF(TA*TB.LE.0.0)GO TO 570
      TZ = (TA-TB) / ALOG(TA/TB)
      GO TO 575
  570 TZ = TA
  575 HS = K - 1.
      IF(PB*PA.LE.0.0)GO TO 576
      HS = K - 1. + 0.001*R*TZ*ALOG(PB/PA)/G
  576 KM = K - 2
      IF(ABS(K-1-HS).GT.0.1) GO TO 578
      GAM=TB-T(I,K+1)
      IF(ABS(GAM).LE.0.01)GOTO 590
      GO TO 582
  578 IF(ABS(TA-TB).LE.0.01)GOTO 590
  580 GAM=(TA-TB)/(K-1-HS)
  582 KM1=KM+1
      IF(ABS(GAM).GT.G) GAM=SIGN(G,GAM)
      DO 585 JD=1,KM1,1
      J=JD-1
      TJ=TA-GAM*(J-HS)
      PJ=PA*(TJ/TA)**(G/(R*GAM*0.001))
      DJ=PJ/(R*TJ)
      P(I,J+1)=PJ
      D(I,J+1)=DJ
  585 T(I,J+1)=TJ
      GO TO 599
  590 KM1=KM+1
      DO 595 JD=1,KM1,1
      J=JD-1
      TJ=TA
      PJ=PA*EXP(-G*(J-HS)/(R*0.001*TJ))
      DJ=PJ/(R*TJ)
      P(I,J+1)=PJ
      D(I,J+1)=DJ
  595 T(I,J+1)=TJ
      IF(NSAME.EQ.2) NSAME=1
  599 HS=0.
      KOUNT = I
      CALL ADJUST
  600 CONTINUE
      RETURN
      END
      SUBROUTINE GRID4D
      REAL LAT,LON
      COMMON/C4/LAT(16),LON(16),NP,P(16,26),R(16,26),T(16,26),SP(16,26),GRID    4
     $ SR(16,26),ST(16,26),DU1,DU2,DUMMY                                 GRID    5
      COMMON /PDTCOM/ IT,MONTH,DUMMY1(8118)                              GRID    6
C                                                                        GRID    7
C                                                                        GRID    8
C     SUBROUTINE TO SELECT PRESSURE, TEMPERATURE, AND DENSITY PROFILES (GRID    9
C     TOGETHER WITH THE NORMALIZED VARIANCES IN EACH, AT UP TO 16  GRID GRID   10
C     AT LAT/LONS SELECTED BY CALLING PROGRAM.                           GRID   11
C                                                                        GRID   12
C     USES NASA HUNTSVILLE MSFC 4-D DATA FILES                           GRID   13
C                                                                        GRID   14
      DIMENSION IN(215)
C                                                                        GRID   16
      COMMON /IOTEMP/ IOTEM1,IOTEM2,DUMMY2(62)                           GRID   17
      COMMON /POINT/ IPT(16,5),LL(16),DXY(16,2)                          GRID   18
      COMMON /ORDER/ IPTN(16,5),IREAD(65,3)                              GRID   19
      COMMON /INT/ D(208,5),IG(5),DYX(2),DLA(4),DLO(4)                   GRID   20
C                                                                        GRID   21
      INTEGER IOTEM1                                                     GRID   22
      CHARACTER*6  M
      CHARACTER*2  CMONTH
C                                                                        GRID   23
      WRITE (CMONTH,'(I2)') MONTH
      IF(MONTH.LT.10)THEN
        OPEN(UNIT=IT,FILE='M'
     1        //CMONTH(2:2)//'.DAT',RECL=213,STATUS='OLD',
     2        FORM='UNFORMATTED')
        ELSE
        OPEN(UNIT=IT,FILE='M'
     1        //CMONTH//'.DAT',RECL=213,STATUS='OLD',
     2        FORM='UNFORMATTED')
        ENDIF
C                                                                        GRID   24
C     INITIALIZE                                                         GRID   25
C                                                                        GRID   26
      ZERO=0.0                                                           GRID   27
      ONE=1.0                                                            GRID   28
      TEN=10.0                                                           GRID   29
      HUNDR=100.0                                                        GRID   30
      THOU=1000.0                                                        GRID   31
C                                                                        GRID   34
      N=MONTH-1-((2*MONTH)/9)*4                                          GRID   35
      IF(MONTH.EQ.13) N=0                                                GRID   36
      NUMEOF = 0                                                         GRID   37
C                                                                        GRID   41
C     CAUTION :  APPROPRIATE 4-D INPUT DATA FILE MUST BE ASSIGNED TO     GRID   42
C                PROPER UNIT BEFORE READING IN DATA.                     GRID   43
C                                                                        GRID   44
      REWIND (IT)
C
   20 CALL SELEC4                                                        GRID   45
C                                                                        GRID   46
      IRC=0                                                              GRID   47
      IRN=1                                                              GRID   48
      IF(IREAD(IRN,3).EQ.0) GO TO 39                                     GRID   49
   21 JT=IT                                                              GRID   50
      M=' READ '                                                         GRID   51
   22 READ(IT,ERR=50,END=39) (IN(I),I=1,213)                             GRID   52
      IRC =IRC +1                                                        GRID   53
      GO TO 60
   50 WRITE(6,23) IT, IRC                                                GRID   55
   23 FORMAT(1H ,' INPUT UNIT NO.',I3,' IN ERROR FOR RECORD NO.  ',I5)
   60 CONTINUE
      IF(IRC.LT.IREAD(IRN,3)) GO TO 22                                   GRID   58
      IF(IRC.GT.IREAD(IRN,3)) GO TO 39                                  GRID  59
   24 I=IREAD(IRN,1)                                                    GRID  60
      J=IREAD(IRN,2)                                                    GRID  61
      IF(IRN.EQ.1) GO TO 25                                             GRID  62
      IF(IREAD(IRN,3).EQ.IREAD(IRN-1,3)) GO TO 27                       GRID  63
   25 IP=IN(212)                                                        GRID  64
      MP=IN(213)                                                        GRID  65
      IF((MP.NE.MONTH).OR.(IP.NE.IPT(I,J))) GO TO 39                    GRID  66
      DO 26 K=213,1,-1                                                  GRID  67
      IN(K+2)=IN(K)                                                     GRID  69
   26 CONTINUE
   27 IN(1) = I
      IN(2) = J                                                         GRID  72
      JT=IOTEM1                                                         GRID  73
      M=' WRITE'                                                        GRID  74
      WRITE(IOTEM1) IN                                                  GRID  75
      IRN=IRN+1                                                         GRID  76
      IF(IREAD(IRN,3).EQ.IRC) GO TO 24                                  GRID  77
      IF(IREAD(IRN,3).EQ.0) GO TO 28                                    GRID  78
      GO TO 21                                                          GRID  79
C                                                                       GRID  80
C     INTERPOLATE TO GIVEN LAT/LON FROM GRID DATA                       GRID  81
C                                                                       GRID  82
   28 M=' READ '                                                        GRID  83
      DO 38 II=1,NP                                                     GRID  84
      DO 29 I=1,208                                                     GRID  85
      DO 29 J=1,5                                                       GRID  86
      D(I,J)=0.0                                                        GRID  87
   29 CONTINUE                                                          GRID  88
      DO 32 J=1,4                                                       GRID  89
      IF(IPT(II,J).EQ.0) GO TO 32                                       GRID  90
      INDEX1 = II
      INDEX2 = J
      REWIND (IOTEM1)                                                   GRID  93
30    READ(IOTEM1,END=39) IN                                            GRID  94
      IF(IN(1) .NE. INDEX1  .OR.  IN(2) .NE. INDEX2) GO TO 30           GRID  95
      DO 31 I=3,210                                                     GRID  96
      D(I-2,J) = IN(I)/HUNDR                                            GRID  99
   31 CONTINUE                                                          GRID 101
      DLA(J) = IN(211)/TEN                                              GRID 102
      DLO(J) = IN(212)/TEN                                              GRID 103
   32 CONTINUE                                                          GRID 104
C                                                                       GRID 105
C     IF NECESSARY, INTERPOLATE                                         GRID 106
C                                                                       GRID 107
      LALO=LL(II)                                                       GRID 108
      DO 33 I=1,5                                                       GRID 109
      IG(I)=IPT(II,I)                                                   GRID 110
   33 CONTINUE                                                          GRID 111
      IF(IG(2).NE.0) GO TO 35                                           GRID 112
      DO 34 I=1,208                                                     GRID 113
      D(I,5)=D(I,1)                                                     GRID 114
   34 CONTINUE                                                          GRID 115
      GO TO 37                                                          GRID 116
   35 IF(IG(5).NE.2) GO TO 36                                           GRID 117
      DYX(1)=DXY(II,1)                                                  GRID 118
      DYX(2)=DXY(II,2)                                                  GRID 119
C                                                                       GRID 120
   36 CALL INTRP4 (LALO)                                                GRID 121
C                                                                       GRID 122
   37 DO 38 I=1,26                                                      GRID 123
      P(II,I)=D(I,5)*HUNDR                                              GRID 124
      T(II,I)=D(I+156,5)/THOU                                           GRID 125
      1(II,I) =D(I+52,5)                                                GRID 126
      DIVIDE=ONE                                                        GRID 127
      IF(P(II,I).GT.ZERO) DIVIDE=(P(II,I)/HUNDR)**2                     GRID 128
      SP(II,I)=D(I+26,5)/DIVIDE                                         GRID 129
      DIVIDE=ONE                                                        GRID 130
      IF(R(II,I).GT.ZERO) DIVIDE=(THOU*R(II,I))**2                      GRID 131
      SR(II,I)=D(I+182,5)/DIVIDE                                        GRID 132
      DIVIDE=ONE                                                        GRID 133
      IF(T(II,I).GT.ZERO) DIVIDE=T(II,I)**2                             GRID 134
      ST(II,I)=D(I+78,5)/DIVIDE                                         GRID 135
   38 CONTINUE                                                          GRID 136
      REWIND (IOTEM1)                                                   GRID136B
      RETURN                                                            GRID 137
   39 I = IREAD(IRN,1)                                                  GRID137B
      J = IREAD(IRN,2)                                                  GRID137C
      WRITE(6,40) JT,IRC,IREAD(IRN,3),MP,MONTH,IP,I,J,IPT(I,J),IRN,M    GRID 138
   40 FORMAT(1X,'***** UNIT NO. ',I3,' IN ERROR ',I7,' RECORDS READ'/
     1 'IREAD(IRN,3) =',I5,' MP = ',I3,' MONTH = ',I3,' IP = ',I5,
     2 ' IPT(',I2,',',I1,') = ',I5,' IRN =',I3/' STATUS: ',A6)
      STOP 'EXECUTION TERMINATED DUE TO ERROR CONDITION'                GRID 142
      END                                                               GRID 143
      SUBROUTINE ADJUST                                                 ADJU   1
      COMMON/C4/DUN1(32),NG,P(16,26),D(16,26),T(16,26),SP(16,26)        ADJU   2
     $,SD(16,26),ST(16,26),DU1,DU2,HS                                   ADJU   3
      COMMON/ADJCOM/A(26,3), B(26), X(26), KOUNT                        ADJU   4
      DIMENSION PQ(26), QQ(26), UC(26), VC(26), WC(26), U(26), V(26),   ADJU   5
     $ W(26)                                                            ADJU   6
C     ASSUMPTIONS@                                                      ADJU   7
C       HS IS THE SURFACE LEVEL                                         ADJU   8
C       ALL DATA VALUES ABOVE SURFACE LEVEL ARE IN 1 KM INCREMENTS      ADJU   9
      E1=0.075                                                          ADJU  10
      E2=0.150                                                          ADJU  11
      MAXIT=3                                                           ADJU  12
      KSMAX=10                                                          ADJU  13
      HSJ = HS                                                          ADJU  14
      IF (HS.LT.0.) HSJ = 0.                                            ADJU  15
      JJ=INT(HSJ+2.)                                                    ADJU  16
      STEST=0.05                                                        ADJU  17
      ISS=1                                                             ADJU  18
      CONST=28703./980.665                                              ADJU  19
      N=26                                                              ADJU  20
      ITER=0                                                            ADJU  21
      UC(1)=SQRT(ABS(SP(KOUNT,1)))                                      ADJU  22
      VC(1)=SQRT(ABS(SD(KOUNT,1)))                                      ADJU  23
      WC(1)=SQRT(ABS(ST(KOUNT,1)))                                      ADJU  24
      DO 5 I=JJ,N                                                       ADJU  25
      UC(I)=SQRT(ABS(SP(KOUNT,I)))                                      ADJU  26
      VC(I)=SQRT(ABS(SD(KOUNT,I)))                                      ADJU  27
    5 WC(I)=SQRT(ABS(ST(KOUNT,I)))                                      ADJU  28
      NM=N-1                                                            ADJU  29
      NP=N+1                                                            ADJU  30
C.....SETS UP QUADRATURE FACTORS                                        ADJU  31
      PQ(1)=500.*(FLOAT(INT(HSJ+1.))-HS)/(CONST*T(KOUNT,1))             ADJU  32
      QQ(1)=500.*(FLOAT(INT(HSJ+1.))-HS)/(CONST*T(KOUNT,JJ))            ADJU  33
      DO 15 I=JJ,NM                                                     ADJU  34
      IP=I+1                                                            ADJU  35
      PQ(I)=500./(CONST*T(KOUNT,I))                                     ADJU  36
   15 QQ(I)=500./(CONST*T(KOUNT,IP))                                    ADJU  37
      GO TO 58                                                          ADJU  38
   12 NM=N-1                                                            ADJU  39
      NP=N+1                                                            ADJU  40
      DO 14 I=1,26                                                      ADJU  41
      U(I)=UC(I)*UC(I)                                                  ADJU  42
      V(I)=VC(I)*VC(I)                                                  ADJU  43
      W(I)=WC(I)*WC(I)                                                  ADJU  44
   14 CONTINUE                                                          ADJU  45
C.....INITIALIZE A(I,J)                                                 ADJU  46
      DO 20 I=1,26                                                      ADJU  47
      DO 20 J=1,3                                                       ADJU  48
   20 A(I,J)=0.                                                         ADJU  49
C.....SETS UP COEFFICIENTS                                              ADJU  50
      I2=0                                                              ADJU  51
      DO 35 I=1,NM                                                      ADJU  52
      IF(I.GT.1.AND.I.LT.JJ) GO TO 35                                   ADJU  53
      AW=1./SP(KOUNT,I)                                                 ADJU  54
      BW=1./SD(KOUNT,I)                                                 ADJU  55
      CW=1./ST(KOUNT,I)                                                 ADJU  56
      IM=I-1                                                            ADJU  57
      IF(I.EQ.JJ) IM=1                                                  ADJU  58
      IP=I+1                                                            ADJU  59
      IF (I.EQ.1) IP=JJ                                                 ADJU  60
      I2=I2+1                                                           ADJU  61
      AW1=1./SP(KOUNT,IP)                                               ADJU  62
      BW1=1./SD(KOUNT,IP)                                               ADJU  63
      CW1=1./ST(KOUNT,IP)                                               ADJU  64
      IF(I.EQ.1) GO TO 25                                               ADJU  65
      A(I2,1)=-(1.-QQ(IM))*(1.+PQ(I))/AW+(1./BW+1./CW)*PQ(I)*QQ(IM)     ADJU  66
   25 A(I2,2)=((1.-QQ(I))**2)/AW1+((1.+PQ(I))**2)/AW+(1./BW+1./CW)      ADJU  67
     $ *(PQ(I)**2)+(1./BW1+1./CW1)*QQ(I)**2                             ADJU  68
      IF(I.EQ.NM) GO TO 30                                              ADJU  69
      A(I2,3)=-(1.-QQ(I))*(1.+PQ(IP))/AW1+(1./BW1+1./CW1)*              ADJU  70
     $ PQ(IP)*QQ(IP)                                                    ADJU  71
   30 B(I2)=U(IP)-U(I)-(U(I)-V(I)+W(I))*PQ(I)-(U(IP)-V(IP)+W(IP))*QQ(I) ADJU  72
   35 CONTINUE                                                          ADJU  73
      CALL DIAGEQ(I2)                                                   ADJU  74
C.....FINDS CORRECTIONS                                                 ADJU  75
      AW=1./SP(KOUNT,1)                                                 ADJU  76
      BW=1./SD(KOUNT,1)                                                 ADJU  77
      CW=1./ST(KOUNT,1)                                                 ADJU  78
      UC(1)=SQRT(ABS(U(1)+X(1)*(1.+PQ(1))/AW))                          ADJU  79
      VC(1)=SQRT(ABS(V(1)-X(1)*PQ(1)/BW))                               ADJU  80
      WC(1)=SQRT(ABS(W(1)+X(1)*PQ(1)/CW))                               ADJU  81
      AW=1./SP(KOUNT,N)                                                 ADJU  82
      BW=1./SD(KOUNT,N)                                                 ADJU  83
      CW=1./ST(KOUNT,N)                                                 ADJU  84
      UC(N)=SQRT(ABS(U(N)-X(I2)*(1.-QQ(NM))/AW))                        ADJU  85
      VC(N)=SQRT(ABS(V(N)-X(I2)*QQ(NM)/BW))                             ADJU  86
      WC(N)=SQRT(ABS(W(N)+X(I2)*QQ(NM)/CW))                             ADJU  87
      I2=1                                                              ADJU  88
      DO 40 I=JJ,NM                                                     ADJU  89
      I2=I2+1                                                           ADJU  90
      I2M=I2-1                                                          ADJU  91
      AW=1./SP(KOUNT,I)                                                 ADJU  92
      BW=1./SD(KOUNT,I)                                                 ADJU  93
      CW=1./ST(KOUNT,I)                                                 ADJU  94
      IM=I-1                                                            ADJU  95
      IF(I.EQ.JJ)IM=1                                                   ADJU  96
      UC(I)=ABS(U(I)        +(-X(I2M)*(1.-QQ(IM))+X(I2)*(1.+PQ(I)))/AW) ADJU  97
      UC(I)=SQRT(UC(I))                                                 ADJU  98
      VC(I)=ABS(V(I)        -(X(I2M)*QQ(IM)+X(I2)*PQ(I))/BW)            ADJU  99
      VC(I)=SQRT(VC(I))                                                 ADJU 100
      WC(I)=ABS(W(I)        +(X(I2M)*QQ(IM)+X(I2)*PQ(I))/CW)            ADJU 101
   40 WC(1)=SQRT(WC(I))                                                 ADJU 102
C.....GETS ADJUSTED VALUES                                              ADJU 103
C.....   ADJUSTS ON TRIANGLE INEQUALITIES                               ADJU 104
   58 K=0                                                               ADJU 105
      DO 68 I=1,N                                                       ADJU 106
      IF(I GT.1.AND.I.LT.JJ) GO TO 68                                   ADJU 107
      AU=UC(I)                                                          ADJU 108
      AV=VC(I)                                          ADJU 109
      AM=WC(I)                                          ADJU 110
      AMAX=AMAX1(AU,AV,AM)                              ADJU 111
      EE=E1*AMAX                                        ADJU 112
      EF=E2*AMAX                                        ADJU 113
      AW=SP(KOUNT,I)                                    ADJU 114
      BW=SD(KOUNT,I)                                    ADJU 115
      CW=ST(KOUNT,I)                                    ADJU 116
      COR=AU+AV-AM-EE                                   ADJU 117
      DIV=AW+BW+CW                                      ADJU 118
      IF(COR.GT.0.) GO TO 60                            ADJU 119
      COR=(AU+AV-AM-EF)/DIV                             ADJU 120
      AU=AU-COR*AW                                      ADJU 121
      AV=AV-COR*BW                                      ADJU 122
      AM=AM+COR*CW                                      ADJU 123
      GO TO 64                                          ADJU 124
   60 COR=AU-AV+AM-EE                                   ADJU 125
      IF(COR.GT.0.) GO TO 62                            ADJU 126
      COR=(AU-AV+AM-EF)/DIV                             ADJU 127
      AU=AU-COR*AW                                      ADJU 128
      AV=AV+COR*BW                                      ADJU 129
      AM=AM-COR*CW                                      ADJU 130
      GO TO 64                                          ADJU 131
   62 COR=-AU+AV+AM-EE                                  ADJU 132
      IF(COR.GT.0.) GO TO 66                            ADJU 133
      COR=(-AU+AV+AM-EF)/DIV                            ADJU 134
      AU=AU+COR*AW                                      ADJU 135
      AV=AV-COR*BW                                      ADJU 136
      AM=AM-COR*CW                                      ADJU 137
   64 K=K+1                                             ADJU 138
   66 UC(I)=AU                                          ADJU 139
      VC(I)=AV                                          ADJU 140
      WC(I)=AM                                          ADJU 141
   68 CONTINUE                                          ADJU 142
      KMAX=K                                            ADJU 143
  100 IF((ITER.EQ.0).OR.(KMAX.NE.0)) GO TO 110          ADJU 144
      GO TO 112                                         ADJU 145
  110 ITER=ITER+1                                       ADJU 146
      IF(ITER.LE.MAXIT) GO TO 12                        ADJU 147
  112 IF (ISS.NE.1) GO TO 999                           ADJU 148
  114 ITER=1                                            ADJU 149
      ISS=2                                             ADJU 150
      VTA=VC(1)                                         ADJU 151
      WTA=WC(1)                                         ADJU 152
      DO 120 I=JJ,NM                                    ADJU 153
      IM=I-1                                            ADJU 154
      IF(I.EQ.JJ)IM=1                                   ADJU 155
      VTB=VC(I)                                         ADJU 156
      WTB=WC(I)                                         ADJU 157
      VC(I)=(VC(I+1)+2.*VTB+VTA)*0.25                   ADJU 158
      WC(I)=(WC(I+1)+2.*WTB+WTA)*0.25                   ADJU 159
      VTA=VTB                                           ADJU 160
      WTA=WTB                                           ADJU 161
  120 CONTINUE                                          ADJU 162
      GO TO 12                                          ADJU 163
C.....CALCULATE THE CORRECTED VARIANCES                  ADJU 164
  999 DO 1010 I=1,N                                     ADJU 165
      IF(I.GT.1.AND.I.LT.JJ) GO TO 1010                 ADJU 166
      SP(KOUNT,I)=UC(I)**2                              ADJU 167
      SD(KOUNT,I)=VC(I)**2                              ADJU 168
      ST(KOUNT,I)=WC(I)**2                              ADJU 169
 1010 CONTINUE                                          ADJU 170
      RETURN                                            ADJU 171
      END                                               ADJU 172
      SUBROUTINE SELEC4                                                 SELE   1
      INTEGER IOTEM2                                                    SELE   2
      COMMON/C4/XL(16),YL(16),NP,DUMMY(2499)                            SELE   3
C                                                                       SELE   4
C     S                                                                 SELE   5
C     SUBROUTINE TO SELECT POINTS FOR INTERPOLATION                     SELE   6
C                                                                       SELE   7
      COMMON /IOTEMP/ IOTEM1,IOTEM2,DUMMY2(62)                          SELE   8
      COMMON /POINT/ IPT(16,5),LL(16),DXY(16,2)                         SELE   9
      COMMON /ORDER/ IPTN(16,5),IREAD(65,3)                             SELE  10
C                                                                       SELE  11
      DIMENSION               IC(4),IL(2),JL(2) LIML(51),LIMU(51)       SELE  12
C                                                                       SELE  13
      DATA  LIML/15,14,13,12,11,10,9,8,7,6,5,4,3,2,23*1,2,3,4,5,6,7,8,9,SELE  14
     110,11,12,13,14,15/                                                SELE  15
      DATA  LIMU/33,34,35,36,37,38,39,40,41,42,43,44,45,46,23*47,46,45, SELE  16
     144,43,42,41,40,39,38,37,36,35,34,33/                              SELE  17
      DATA PI/3.14159/                                                  SELE  18
C                                                                       SELE  19
C                                                                       SELE  20
C        INITIALIZE                                                     SELE  21
C                                                                       SELE  22
      PI4=PI/4.                                                         SELE  23
      DEGRAD=PI/180.                                                    SELE  24
      DO 1 I=1,16                                                       SELE  25
      DO 1 J=1,5                                                        SELE  26
    1 IPT(I,J)=0                                                        SELE  27
C                                                                       SELE  28
C        MAJOR LOOP FOR POINTS                                          SELE  29
C                                                                       SELE  30
      DO 100 II=1,NP                                                    SELE  31
C                                                                       SELE  32
      LA=ABS(XL(II))*10.+.5                                             SELE  33
      LO=YL(II)*10.+.5                                                  SELE  34
      IF (LO.LT.0) LO = LO + 3600                                       SELE 34B
      LL(II)=LA*10000+LO                                                SELE  35
      IF (XL(II).LT.0.) LL(II)=-LL(II)                                  SELE  36
C                                                                       SELE  37
      IF (XL(II)-15.1) 15,30,30                                         SELE  38
   15 IF (XL(II)) 50,40,40                                              SELE  39
C                                                                       SELE  40
C        NMC GRID                                                       SELE  41
C                                                                       SELE  42
   30 IPT(II,5)=2                                                       SELE  43
      YEL = YL(II)                                                      SELE 43B
      IF (YEL.LT.0.) YEL = YEL + 360.                                   SELE 43C
      EL=(350.-YEL)*DEGRAD                                              SELE  44
      PHI=XL(II)*DEGRAD                                                 SELE  45
      R=31.204359052*(SIN(PI4-PHI/2.)/COS(PI4-PHI/2.))                  SELE  46
      XX=R*COS(EL)+24.                                                  SELE  47
      YY=R*SIN(EL)+26.                                                  SELE  48
      I=XX                                                              SELE  49
      J=YY                                                              SELE  50
      DX=XX-I                                                           SELE  51
      DY=YY-J                                                           SELE  52
      DXY(II,1)=DX                                                      SELE  53
      DXY(II,2)=DY                                                      SELE  54
      IF (XL(II).GT.17.18) GO TO 31                                     SELE  55
      IF ((J.LT.1).OR.(J.GT.51)) GO TO 70                               SELE  56
      IF ((I.LT.LIML(J)).OR.(I.GT.LIMU(J))) GO TO 70                    SELE  57
   31 IC(1)=I*1000+J                                                    SELE  58
      IF ((ABS(DX).GT..1).OR.(ABS(DY).GT..1)) GO TO 32                  SELE  59
      IP=1                                                              SELE  60
      GO TO 35                                                          SELE  61
   32 CONTINUE                                                          SELE  62
      IF (XL(II).GT.17.18) GO TO 34                                     SELE  63
      IF (((I.GT.(LIMU(J)-1)).AND.((J.GE.15).AND.(J.LE.37)))            SELE  64
     1 .OR.(J.GT.50)) GO TO 70                                          SELE  65
      IF ((I+1.GT.LIMU(J+1)).OR.(I.LT.LIML(J+1))) GO TO 80              SELE  66
      IF ((I.EQ.LIMU(J)).OR.(I.EQ.LIML(J))) GO TO 80                    SELE  67
   34 IP=4                                                              SELE  68
      IC(2)=(I+1)*1000+J                                                SELE  69
      IC(3)=I*1000+J+1                                                  SELE  70
      IC(4)=(I+1)*1000+J+1                                              SELE  71
   35 CONTINUE                                                          SELE  72
      REWIND (IOTEM2)                                                   SELE  73
      DO 38 IPG=1,1977                                                  SELE  74
      READ(IOTEM2) IJ                                                   SELE  75
      DO 38 K=1,IP                                                      SELE  76
   38 IF(IC(K).EQ.IJ) IPT(II,K)=IPG                                     SELE  77
      GO TO 100                                                         SELE  78
C                                                                       SELE  79
C         EQUATORIAL GRID                                               SELE  80
C                                                                       SELE  81
   40 IPT(II,5)=1                                                       SELE  82
      L1=XL(II)                                                         SELE  83
      L2=YL(II)                                                         SELE  84
      IF (L2.LT.0) L2 = L2 + 360                                        SELE 84B
      IL(1)=L1/5                                                        SELE  85
      IL(2)=IL(1)+1                                                     SELE  86
      JL(1)=(L2/5)+1                                                    SELE  87
      JL(2)=JL(1)-1                                                     SELE  88
      DO 45 K1=1,2                                                      SELE  89
      DO 45 K2=1.2                                                      SELE  90
      IF ((APS(XL(II)-IL(K1)*5).GT.0.1).OR.(ABS(YL(II)-JL(K2)*5).GT.0.1)SELE  91
     1 ) GO TO 45                                                       SELE  92
      IF (JL(K2).EQ.72) JL(K2)=0                                        SELE  93
      IPT(II,1)=JL(K2)*4+IL(K1)+1                                       SELE  94
      GO TO 100                                                         SELE  95
   45 CONTINUE                                                          SELE  96
      IF (JL(1).EQ.72) JL(1)=0                                          SELE  97
      IPT(II,1)=JL(1)*4+IL(1)+1                                         SELE  98
      IPT(II,2)=JL(2)*4+IL(1)+1                                         SELE  99
      IPT(II,3)=JL(1)*4+IL(2)+1                                         SELE 100
      IPT(II,4)=JL(2)*4+IL(2)+1                                         SELE 101
      GO TO 100                                                         SELE 102
C                                                                       SELE 103
C         SOUTHERN HEMISPHERE                                           SELE 104
C                                                                       SELE 105
   50 IPT(II,5)=3                                                       SELE 106
      L1=XL(II)                                                         SELE 107
      L2=YL(II)                                                         SELE 108
      IF (L2.LT.0) L2 = L2 + 360                                        SELE108B
      IF (ABS(XL(II)).LT.85.0) GO TO 51                                 SELE 109
      IPT(II,1)=1                                                       SELE 110
      IF (ABS(XL(II)+90.).LT.0.11) GO TO 100                            SELE 111
   51 CONTINUE                                                          SELE 112
      IL(1)=(L1/5)-1                                                    SELE 113
      JL(1)=(L2/5)+1                                                    SELE 114
      IL(2)=IL(1)+1                                                     SELE 115
      JL(2)=JL(1)-1                                                     SELE 116
      DO 52 K1=1,2                                                      SELE 117
      DO 52 K2=1,2                                                      SELE 118
      IF ((ABS(XL(II)-IL(K1)*5).GT.0.1).OR.(ABS(YL(II)-JL(K2)*5).GT.0.1)SELE 119
     1 ) GO TO 52                                                       SELE 120
      IF (JL(K2).EQ.72) JL(K2)=0                                        SELE 121
      IPT(II,1)=JL(K2)*17-IL(K1)+1                                      SELE 122
      IF (IL(K1).NE.0) GO TO 100                                        SELE 123
      IPT(II,1)=JL(K2)*4+1                            SELE 124
      IPT(II,5)=1                                     SELE 125
      GO TO 100                                       SELE 126
   52 CONTINUE                                        SELE 127
      IF (JL(1).EQ.72) JL(1)=0                        SELE 128
      IF (IPT(II,1).EQ.1) GO TO 54                    SELE 129
      IPT(II,1)=JL(1)*17-IL(1)+1                      SELE 130
      IPT(II,2)=JL(2)*17-IL(1)+1                      SELE 131
      IF (IL(2)) 55,53,55                             SELE 132
   53 IPT(II,3)=JL(1)*4+1                             SELE 133
      IPT(II,4)=JL(2)*4+1                             SELE 134
      IPT(II,5)=1133                                  SELE 135
      GO TO 100                                       SELE 136
   54 IPT(II,2)=JL(1)*17-IL(2)+1                      SELE 137
      IPT(II,3)=JL(2)*17-IL(2)+1                      SELE 138
      IPT(II,5)=333                                   SELE 139
      GO TO 100                                       SELE 140
   55 CONTINUE                                        SELE 141
      IPT(II,3)=JL(1)*17-IL(2)+1                      SELE 142
      IPT(II,4)=JL(2)*17-IL(2)+1                      SELE 143
      GO TO 100                                       SELE 144
C                                                     SELE 145
C     BODERLINE POINTS                                SELE 146
C                                                     SELE 147
   70 CONTINUE                                        SELE 148
C     TWO NMC, TWO EQUATORIAL                         SELE 149
      IPT(II,5)=2211                                  SELE 150
      L=YL(II)                                        SELE 151
      IPT(II,1)=((L/5)+2)*4                           SELE 152
      IPT(II,2)=IPT(II,1)-4                           SELE 153
      IF (L.GE.355) IPT(II,1)=4                       SELE 154
C                                                     SELE 155
      IF (J.LT.1) J=1                                 SELE 156
      IF (J.GT.51) J=51                               SELE 157
      IF (I.LT.LIML(J)) I=LIML(J)                     SELE 158
      IF (I.GT.LIMU(J)) I=LIMU(J)                     SELE 159
      IC(1)=I*1000+J                                  SELE 160
      IF ((J.LT.15).OR.(J.GT.37)) GO TO 72            SELE 161
      IC(2)=I*1000+J+1                                SELE 162
      GO TO 76                                        SELE 163
   72 IF ((J.NE.1).AND.(J.NE.51)) GO TO 74            SELE 164
      IF (I.EQ.LIMU(J)) GO TO 73                      SELE 165
      IC(2)=(I+1)*1000+J                              SELE 166
      GO TO 76                                        SELE 167
   73 IC(2)=(I-1)*1000+J                              SELE 168
      GO TO 76                                        SELE 169
   74 IF (I.EQ.LIML(J)) GO TO 75                      SELE 170
      IC(2)=LIMU(J+1)*1000+J+1                        SELE 171
      GO TO 76                                        SELE 172
   75 IC(2)=LIML(J+1)*1000+J+1                        SELE 173
C                                                     SELE 174
   76 REWIND (IOTEM2)                                 SELE 175
      DO 77 IPG=1,1977                                SELE 176
      READ(IOTEM2) IJ                                 SELE 177
      DO 77 K=1,2                                     SELE 178
   77 IF(IC(K).EQ.IJ) IPT(II,K+2)=IPG                 SELE 179
      GO TO 100                                       SELE 180
C                                                     SELE 181
   80 CONTINUE                                        SELE 182
C     THREE NMC, ONE EQUATORIAL                       SELE 183
      IPT(II,5)=2212                                  SELE 184
      IC(2) = 0                                       SELE 185
      L=YL(II)                                        SELE 186
      IPT(II,2)=((L/5)+1)*4                           SELE 187
      IF (L.GE.355) IPT(II,2)=4                                   SELE 188
      IF (I.EQ.LIML(J)) GO TO 84                                  SELE 189
      IF (J.GT.37) GO TO 82                                       SELE 190
      IC(1)=I*1000+J                                              SELE 191
      IC(3)=I*1000+J+1                                            SELE 192
      IC(4)=(I+1)*1000+J+1                                        SELE 193
      GO TO 88                                                    SELE 194
   82 IC(1)=(I+1)*1000+J                                          SELE 195
      IC(3)=I*1000+J                                              SELE 196
      IC(4)=I*1000+J+1                                            SELE 197
      GO TO 88                                                    SELE 198
   84 IF (J.GT.37) GO TO 86                                       SELE 199
      IC(1)=(I-1)*1000+J+1                                        SELE 200
      IC(3)=I*1000+J+1                                            SELE 201
      IC(4)=I*1000+J                                              SELE 202
      GO TO 88                                                    SELE 203
   86 IC(1)=(I+1)*1000+J+1                                        SELE 204
      IC(3)=(I+1)*1000+J                                          SELE 205
      IC(4)=I*1000+J                                              SELE 206
C                                                                 SELE 207
   88 REWIND (IOTEM2)                                             SELE 208
      DO 89 IPG=1,1977                                            SELE 209
      READ(IOTEM2) IJ                                             SELE 210
      DO 89 K=1,4                                                 SELE 211
      IF(IC(K).EQ.0) GO TO 89                                     SELE 212
      IF(IC(K).EQ.IJ) IPT(II,K)=IPG                               SELE 213
   89 CONTINUE                                                    SELE 214
C                                                                 SELE 215
  100 CONTINUE                                                    SELE 216
      DO 150 I=1,16                                               SELE 217
      DO 150 J=1,5                                                SELE 218
  150 IPTN(I,J)=IPT(I,J)                                          SELE 219
      CALL SORT4(NP)                                              SELE 220
      RETURN                                                      SELE 221
      END                                                         SELE 222
      SUBROUTINE INTRP4 (LALON)                                   INTR   1
C                                                                 INTR   2
C         SUBROUTINE TO INTERPOLATE VALUES                        INTR   3
C                                                                 INTR   4
      DIMENSION XLL(4),YLL(4),XC(4),YC(4)                         INTR   5
C                                                                 INTR   6
      COMMON/INT/D(208,5),IG(5),DXY(2),DLA(4),DLO(4)              INTR   7
C                                                                 INTR   8
      DEGRAD=3.14159/180.                                         INTR   9
      LALO=IABS(LALON)                                            INTR  10
      L1=LALO/10000                                               INTR  11
      L2=LALO-L1*10000                                            INTR  12
      XL=L1/10.                                                   INTR  13
      YL=L2/10.                                                   INTR  14
      IF (IG(5)-2) 30,20,10                                       INTR  15
   10 IF (IG(5)-3) 30,30,50                                       INTR  16
C                                                                 INTR  17
C         INTERPOLATE FROM NMC GRID                               INTR  18
C                                                                 INTR  19
   20 CONTINUE                                                    INTR  20
      DO 25 L=1,26                                                INTR  21
      DO 22 J=1,4                                                 INTR  22
   22 IF (D(L,J).LT.0.01) GO TO 25                                INTR  23
      DO 24 K=1,8                                                 INTR  24
      I=(K-1)*26+L                                                INTR  25
      D(I,5)=(1.-DXY(2))*((1.-DXY(1))*D(I,1)+DXY(1)*D(I,2))       INTR  26
     1 +DXY(2)*(((1.-DXY(1))*D(I,3))+DXY(1)*D(I,4))               INTR  27
   24 CONTINUE                                                    INTR  28
   25 CONTINUE                                                    INTR  29
```

```
      RETURN
C
C     INTERPOLATE FROM EQUATION FOR SOUTHERN HEMISPHERE GRID
C
   30 CONTINUE
      DO 32 J=1,2
      XLL(J)=DLA(J)
      YLL(J)=DLO(J)
      IF ((YL.GE.355.).AND.(YLL(J).LT.0.01)) YLL(J)=360.
   32 CONTINUE
      X=(YLL(1)-YL)/5.
      Y=(XL-XLL(1))/5.
      IF (IG(5).EQ.3) Y=-Y
      DO 38 L=1,26
      DO 36 J=1,4
   36 IF (D(L,J).LT.0.01) GO TO 38
      DO 37 K=1,8
      I=(K-1)*26+L
      D(I,5)=D(I,1)+X*(D(I,2)-D(I,1))+Y*(D(I,3)-D(I,1))+X*Y*
     1 (D(I,4)-D(I,3)-D(I,2)+D(I,1))
   37 CONTINUE
   38 CONTINUE
      RETURN
C
C         INTERPOLATE FROM ACROSS GRIDS
C
   50 CONTINUE
      IF (IG(5).NE.1133) GO TO 55
      IG(5)=3
      GO TO 30
   55 CONTINUE
      IF (IG(5).NE.333) GO TO 60
      DLO(1)=(DLO(2)+DLO(3))/2.
      DO 52 I=1,208
   52 D(I,4)=D(I,3)
      DLA(4)=DLA(3)
      DLO(4)=DLO(3)
   60 CONTINUE
      DO 62 I=1,4
      XLL(I)=DLA(I)
      YLL(I)=DLO(I)
      IF ((YL.GT.350.).AND.(YLL(I).LT.0.01)) YLL(I)=360.
   62 CONTINUE
      ITH=0
      X=YLL(1)-YL
      Y=YL-XLL(1)
   63 CONTINUE
      DO 65 I=2,4
      XC(I)=YLL(1)-YLL(I)
   65 YC(I)=XLL(I)-XLL(1)
      TH2=3.14159/4
      TH3=3.14159/4
      IF (ABS(XC(2)).GT.0.01) TH2=ATAN(YC(2)/XC(2))
      IF (ABS(YC(3)).GT.0.01) TH3=ATAN(XC(3)/YC(3))
      IF (XC(2).LT.0.) TH2=3.14159+TH2
      IF (XC(3).LT.0.) TH3=3.14159+TH3
      DNN=COS(TH2+TH3)
      IF (ABS(DNN).GT.0.001) GO TO 66
      ITH=ITH+1
      IF (ITH.EQ.2) GO TO 66
      XLL(3)=XLL(4)
      YLL(3)=YLL(4)
      DO 61 I=1,208
   61 D(I,3)=D(I,4)
```

```
      GO TO 63                                                          INTR  94
   66 CONTINUE                                                          INTR  95
      ZA=SQRT(XC(2)**2+YC(2)**2)                                        INTR  96
      IF (ITH.LT.2) GO TO 69                                            INTR  97
      Z=SQRT(X**2+Y**2)                                                 INTR  98
      E=0.                                                              INTR  99
      Z4=0.                                                             INTR 100
      GO TO 71                                                          INTR 101
   69 CONTINUE                                                          INTR 102
      EB=SQRT(XC(3)**2+YC(3)**2)                                        INTR 103
      Z4=(XC(4)*COS(TH3)-YC(4)*SIN(TH3))/DNN                            INTR 104
      E4=(YC(4)*COS(TH2) XC(4)*SIN(TH2))/DNN                            INTR 105
      Z=(X*COS(TH3)-Y*SIN(TH3))/DNN                                     INTR 106
      E=(Y*COS(TH2)-X*SIN(TH2))/DNN                                     INTR 107
      B=0.                                                              INTR 108
      C=0.                                                              INTR 109
      DD=0.                                                             INTR 110
C                                                                       INTR 111
   71 CONTINUE                                                          INTR 112
      DO 70 L=1,26                                                      INTR 113
      DO 68 J=1,4                                                       INTR 114
   68 IF (D(L,J).LT.0.01) GO TO 70                                      INTR 115
      DO 67 K=1,8                                                       INTR 116
      I=(K-1)*26+L                                                      INTR 117
      A=D(I,1)                                                          INTR 118
      IF (ZA.GT.0.01) B=(D(I,2)-D(I,1))/ZA                              INTR 119
      IF (EB.GT.0.01) C=(D(I,3)-D(I,1))/EB                              INTR 120
      IF ((ABS(Z4).GT.0.01).AND.(ABS(E4).GT.0.01))                      INTR 121
     1 DD=(D(I,4)-A-B*Z4-C*E4)/(Z4*E4)                                  INTR 122
      D(I,5)=A+B*Z+C*E+DD*Z*E                                           INTR 123
   67 CONTINUE                                                          INTR 124
   70 CONTINUE                                                          INTR 125
      RETURN                                                            INTR 126
      END                                                               INTR 127
      SUBROUTINE DIAGEQ(N)                                              DIAG   1
C     A(I,J)=DIAG. TERMS, I=ROW NO., J=DIAG. NO.                        DIAG   2
C     B(I)=RIGHT SIDE TERMS                                             DIAG   3
C     N=NO. OF ROWS                                                     DIAG   4
C     K=NO. OF BORDER DIAGONALS, M=K+1=INDES OF PRIN. DIAG              DIAG   5
C               2KH=TOTAL NO. OF DIAGS.                                 DIAG   6
C     X(I)=SOLUTION                                                     DIAG   7
      COMMON/ADJCOM/A(26,3), B(26), X(26), KOUNT                        DIAG   8
      K = 1                                                             DIAG   9
      M=K+1                                                             DIAG  10
      DO 30 L=1,N                                                       DIAG  11
      ALM=A(L,M)                                                        DIAG  12
      A(L,M)=1.                                                         DIAG  13
      IF(L.EQ.N) GO TO 15                                               DIAG  14
      I2=MIN0(K,N-L)                                                    DIAG  15
      DO 10 I=1,I2                                                      DIAG  16
      MPI=M+I                                                           DIAG  17
   10 A(L MPI)=A(L,MPI)/ALM                                             DIAG  18
   15 B(L)=B(L)/ALM                                                     DIAG  19
      IF(L.EQ.N) GO TO 30                                               DIAG  20
      DO 25 I=1,I2                                                      DIAG  21
      LPI=L+I                                                           DIAG  22
      FACT=A(LPI,M-I)                                                   DIAG  23
      DO 20 J=1,I2                                                      DIAG  24
      MJI=M+J-I                                                         DIAG  25
   20 A(LPI,MJI)=A(LPI,MJI)-A(L,M+J)*FACT                               DIAG  26
   25 B(LPI)=B(LPI)-B(L)*FACT                                           DIAG  27
   30 CONTINUE                                                          DIAG  28
      X(N)=B(N)                                                         DIAG  29
      NM1=N-1                                                           DIAG  30
      DO 50 L=1,NM1                                                     DIAG  31
      NML=N-L                                                           DIAG  32
      SUM=0.                                                            DIAG  33
      I2=MINO(K,L)                                                      DIAG  34
      DO 40 I=1,I2                                                      DIAG  35
   40 SUM=SUM+A(NML,M+I)*X(NML+I)                                       DIAG  36
   50 X(NML)=B(NML)-SUM                                                 DIAG  37
      RETURN                                                            DIAG  38
      END                                                               DIAG  39
      SUBROUTINE SORT4(NP)                                              SORT   1
C                                                                       SORT   2
C                SORTS POINTS FOR SEQUENTIAL TAPE READING               SORT   3
C                                                                       SORT   4
C          ASSIGNS POINT NUMBERS BY ORDER ON TAPE, NOT BY GRID          SORT   5
C                                                                       SORT   6
      COMMON /ORDER/ IPT (16,5),IREAD(65,3)                             SORT   7
C                                                                       SORT   8
      DO 1 I=1,65                                                       SORT   9
      DO 1 J=1,3                                                        SORT  10
    1 IREAD(I,J)=0                                                      SORT  11
      DO 9 I=1,NP                                                       SORT  12
      IF(IPT(I,5).LT.1) GO TO 10                                        SORT  13
      IF(IPT(I,5).EQ.1) GO TO 9                                         SORT  14
      IF(IPT(I,5).EQ.2) GO TO 2                                         SORT  15
      IF(IPT(I,5).EQ.3) GO TO 4                                         SORT  16
      IF(IPT(I,5).EQ.1133)GO TO 6                                       SORT  17
      IF(IPT(I,5).EQ.2211) GO TO 7                                      SORT  18
      IF(IPT(I,5).EQ.2212)GO TO 8                                       SORT  19
      IF (IPT(I,5).EQ.333) GO TO 4                                      SORT  20
      GO TO 10                                                          SORT  21
    2 DO 3 J=1,4                                                        SORT  22
      IF(IPT(I,J).LT.1) GO TO 3                                         SORT  23
      IPT(I,J)=IPT(I,J)+288                                             SORT  24
    3 CONTINUE                                                          SORT  25
      GO TO 9                                                           SORT  26
    4 DO 5 J=1,4                                                        SORT  27
      IF(IPT(I,J).LT.1) GO TO 5                                         SORT  28
      IPT(I,J)=IPT(I,J)+2265                                            SORT  29
    5 CONTINUE                                                          SORT  30
      GO TO 9                                                           SORT  31
    6 IF(IPT(I,1).GT.0)IPT(I,1)=IPT(I,1)+2265                           SORT  32
      IF(IPT(I,2).GT.0)IPT(I,2)=IPT(I,2)+2265                           SORT  33
      GO TO 9                                                           SORT  34
    7 IF(IPT(I,3).GT.0)IPT(I,3)=IPT(I,3)+288                            SORT  35
      IF(IPT(I,4).GT.0)IPT(I,4)=IPT(I,4)+288                            SORT  36
      GO TO 9                                                           SORT  37
    8 IF(IPT(I,1).GT.0)IPT(I,1)=IPT(I,1)+288                            SORT  38
      IF(IPT(I,3).GT.0)IPT(I,3)=IPT(I,3)+288                            SORT  39
      IF(IPT(I,4).GT.0)IPT(I,4)=IPT(I,4)+288                            SORT  40
    9 CONTINUE                                                          SORT  41
C                                                                       SORT  42
C          REORDERS POINT NUMBERS FOR READ                              SORT  43
C                                                                       SORT  44
   10 IP=0                                                              SORT  45
      DO 13 K=1,NP                                                      SORT  46
      DO 13 L=1,4                                                       SORT  47
      MP=IPT(K,L)                                                       SORT  48
      IF(MP.LT.1) GO TO 13                                              SORT  49
   11 II=K                                                              SORT  50
      JJ=L                                                              SORT  51
      DO 12 I=1,NP                                                      SORT  52
      DO 12 J=1,4                                                       SORT  53
      IF (IPT(I,J).LT.1) GO TO 12                                       SORT  54
      IF(IPT(I,J).GT.3490) GO TO 12                                     SORT  55
      IF(IPT(I,J).GE.MP) GO TO 12                                       SORT  56
      II=I                                                              SORT  57
      JJ=J                                                              SORT  58
      MP=IPT(I,J)                                                       SORT  59
   12 CONTINUE                                                          SORT  60
      IF(IPT(II,JJ).GT.3490) GO TO 14                                   SORT  61
      IR=IR+1                                                           SORT  62
      IREAD(IR,1)=II                                                    SORT  63
      IREAD(IR,2)=JJ                                                    SORT  64
      IREAD(IR,3)=IPT(II,JJ)                                            SORT  65
      IPT(II,JJ)=IPT(II,JJ)+9000                                        SORT  66
      MP=IPT(K,L)                                                       SORT  67
      IF(MP.GT.3490) GO TO 13                                           SORT  68
      GO TO 11                                                          SORT  69
   13 CONTINUE                                                          SORT  70
   14 RETURN                                                            SORT  71
      END                                                               SORT  72
!!T72-
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-------------------------------------------------------------------------------
      subroutine GetStringWidth ( ChrStr , lngth , nchar , iwidth , ixchar )
c-------------------------------------------------------------------------------
c     determine the width [iwidth] of the character string [ChrStr] and the
c     distance each character is offset from the beginning [ixchar].  The
c     character string is also left justified.

      integer*2          ixchar(*)
      integer*2          lngth
      integer*2          nchar
      integer*2          iwidth
      character*(*)      ChrStr
      character*1        onechr

c.....determine the number of characters in the string and left justify it

      nchar  = NumChr ( ChrStr , lngth )

c.....determine the width of the string and the individual position of each
c     character

      if ( nchar.gt.0 ) then
         iwidth = 0
         do i = 1 , nchar
            ixchar(i) = iwidth
            onechr    = chrstr(i:i)
            iwidth    = iwidth + CharWidth ( %val(onechr) )
         end do
      else if ( nchar.eq.0 ) then
         iwidth = 0
      end if

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Initialize segment

!!S Initialize
c-------------------------------------------------------------------------------
      subroutine initialize
c-------------------------------------------------------------------------------

!!SETC USINGINCLUDES = FALSE
      implicit none

c.....common block definition files

      include 'AppleMenu.inc'
      include 'FileMenu.inc'
      include 'EditMenu.inc'
      include 'MapMenu.inc'
      include 'MBar.inc'
      include 'Globals.inc'

      integer*2 kSysEnvironsVersion /1/
      integer*2 os_err

c.....declare TrapAvailable routine as a logical function (I wrote it)

      logical TrapAvailable

c.....set up a 1-byte integer for a 2-item enumerated type;
c.....a value of "1" means the second of the two enumerations

      integer*1 ToolTrap /1/

c-------------------------------------------------------------------------------

      gInBackground = .false.

c.....call LSC FORTRAN routine to initialize their data structures

      call InitFORTRAN

      os_err = SysEnvirons ( %ref(kSysEnvironsVersion), %val(gMac) )
      if( gMac.machineType < 0 ) call AlertUser  ! RLH must provide this subroutine

c.....see if WaitNextEvent is available

      gHasWaitNextEvent = TrapAvailable ( TWaitNextEvent, ToolTrap )

c.....set up a handle value in the menuHandle record

      MenuBar.menuH = GetNewMBar ( %val(MenuBarID) )
      call SetMenuBar   ( %val(MenuBar.menuH) )
      call DisposHandle ( %val(MenuBar.menuH) )

c.....set up the menu handles for the menus

      AppleMenuHndl.menuH = GetMHandle ( %val(AppleMenuID) )
      FileMenuHndl.menuH  = GetMHandle ( %val(FileMenuID) )
      EditMenuHndl.menuH  = GetMHandle ( %val(EditMenuID) )
      MapMenuHndl.menuH   = GetMHandle ( %val(MapMenuID) )

c.....add desk accessories to the Apple Menu

      call AddResMenu ( %val( AppleMenuHndl ), %val( 'DRVR' ) )

c.....draw the menu bar

      call DrawMenuBar

      return
      end
```

```
!!s LngDat
c-----------------------------------------------------------------------------
Segment LngDat
      block data LngDat
c-----------------------------------------------------------------------------
c     array of longitude values

      include 'LngCom.inc'

      data (Longitude(i),i=    1,   80)/
     .-128.138,-128.654,-128.769,-129.544,-130.079,-130.481,-130.130,-129.754,
     .-130.129,-130.006,-141.010,-139.591,-138.504,-137.337,-135.975,-134.596,
     .-134.116,-132.957,-132.889,-132.890,-131.579,-130.247,-130.839,-132.135,
     .-132.882,-131.805,-130.525,-129.357,-128.856,-128.856,-127.615,-126.750,
     .-125.934,-125.206,-124.425,-123.125,-122.745,-122.186,-122.578,-122.752,
     .-122.854,-123.628,-123.533,-123.534,-123.847,-124.271,-124.618,-124.860,
     .-125.573,-125.515,-126.161,-126.177,-126.178,-126.972,-127.691,-127.495,
     .-126.688,-127.497,-127.389,-126.984,-127.611,-128.185,-128.139,-130.005,
     .-130.055,-130.327,-130.872,-131.023,-131.916,-131.976,-132.410,-133.169,
     .-133.453,-133.760,-133.769,-133.769,-134.687,-135.131,-135.364,-135.274/
      data (Longitude(i),i=   81,  160)/
     .-135.049,-135.889,-136.489,-136.081,-137.018,-137.769,-138.427,-139.235,
     .-139.852,-139.852,-139.055,-140.043,-141.043,-141.943,-142.946,-143.933,
     .-144.756,-145.703,-146.633,-147.432,-147.726,-147.726,-148.301,-148.125,
     .-148.626,-149.609,-150.297,-151.232,-151.830,-151.467,-151.400,-150.578,
     .-149.550,-149.467,-150.439,-151.355,-151.743,-151.743,-152.315,-152.628,
     .-153.311,-154.122,-153.475,-153.917,-154.326,-155.163,-155.734,-155.737,
     .-156.553,-157.044,-157.864,-158.559,-159.005,-159.684,-160.565,-161.410,
     .-162.187,-163.065,-163.329,-163.329,-162.545,-162.069,-161.261,-160.444,
     .-160.334,-159.623,-158.869,-158.325,-157.459,-157.623,-157.495,-156.981,
     .-156.779,-156.779,-157.462,-158.434,-159.397,-160.300,-161.231,-162.148/
      data (Longitude(i),i=  161,  240)/
     .-161.633,-161.786,-162.208,-162.225,-162.483,-163.378,-164.273,-164.647,
     .-164.648,-164.940,-163.909,-164.879,-165.061,-166.114,-165.731,-165.289,
     .-164.845,-164.122,-163.112,-162.801,-162.803,-161.810,-160.839,-160.945,
     .-160.787,-161.972,-162.768,-163.788,-164.944,-166.109,-166.705,-167.424,
     .-167.846,-167.846,-166.785,-165.582,-164.761,-163.887,-162.678,-161.538,
     .-162.189,-161.054,-161.802,-163.039,-163.883,-164.690,-164.764,-164.756,
     .-165.867,-166.235,-165.072,-163.767,-163.067,-162.515,-161.291,-160.172,
     .-159.770,-159.770,-159.934,-158.446,-157.184,-155.644,-154.586,-153.069,
     .-152.216,-152.216,-151.316,-149.838,-148.441,-147.024,-145.578,-144.098,
     .-142.628,-141.494,-141.010, -67.240, -66.401, -65.335, -64.039, -64.687/
      data (Longitude(i),i=  241,  320)/
     . -63.403, -63.822, -63.816, -62.579, -61.419, -61.887, -63.062, -62.595,
     . -63.500, -63.499, -63.302, -63.511, -64.404, -65.307, -64.798, -65.480,
     . -65.480, -66.731, -67.936, -67.282, -68.447, -68.070, -67.128, -66.736,
     . -66.698, -66.698, -65.715, -65.070, -64.507, -65.079, -65.269, -65.269,
     . -66.379, -67.242, -67.858, -68.992, -68.993, -68.995, -68.359, -67.511,
     . -66.733, -66.149, -67.116, -68.123, -69.121, -69.795, -70.873, -70.829,
     . -70.832, -71.775, -72.195, -73.081, -73.609, -74.622, -74.648, -74.649,
     . -74.673, -75.747, -76.648, -77.785, -78.066, -77.462, -76.264, -75.602,
     . -75.090, -74.716, -74.711, -73.501, -74.079, -73.872, -72.898, -71.671,
     . -71.184, -69.961, -69.227, -69.227, -70.422, -71.664, -72.915, -72.275/
      data (Longitude(i),i=  321,  400)/
     . -72.621, -72.941, -78.501, -77.770, -77.158, -76.658, -76.523, -76.549,
     . -76.760, -77.145, -77.863, -78.076, -78.074, -78.541, -77.884, -77.541,
     . -77.514, -77.641, -77.885, -77.885, -77.743, -78.033, -78.124, -77.162,
     . -76.143, -75.093, -74.023, -73.017, -72.795, -72.798, -71.787  -71.557,
     . -70.580, -69.542, -69.654, -69.622, -70.612, -71.032, -71.031, -70.050,
     . -69.700, -69.483, -68.538, -68.225, -68.794, -69.369, -69.369, -68.536,
     . -68.178, -67.678, -67.035, -66.460, -66.091, -66.023, -65.567, -65.491,
     . -65.488, -64.897, -64.156, -63.871, -63.125, -62.989, -62.990, -63.393,
     . -62.574, -61.991, -61.894, -61.346, -61.346, -62.138, -61.601, -61.145,
     . -60.393, -60.531, -59.777, -59.775, -59.827, -59.260, -58.756, -57.896/
      data (Longitude(i),i=  401,  480)/
     . -58.250, -59.056, -59.769, -60.598, -60.151, -59.458, -59.024, -59.024,
     . -58.372, -57.551, -57.127, -56.307, -55.746, -56.004, -55.793, -56.142,
     . -56.764,-122.747,-121.322,-120.264,-118.907,-117.742,-116.365,-114.978,
     .-114.076,-115.143,-115.114,-115.114,-113.816,-112.497,-111.222,-109.890,
     .-108.927,-107.943,-107.938,-107.200,-107.735,-107.891,-106.757,-106.756,
     .-105.722,-107.051,-108.290,-108.156,-106.855,-105.487,-105.040,-103.857,
     .-102.656,-101.346,-100.588,-100.588, -99.277, -97.957, -98.424, -97.283,
     . -96.747, -96.436, -95.490, -95.803, -95.327, -95.558, -95.443, -88.365,
     . -87.846, -86.635, -86.317, -85.843, -85.198, -85.432, -85.511, -85.584,
     . -85.586, -84.126, -82.727, -82.068, -81.310, -82.254, -82.125, -81.349/
      data (Longitude(i),i=  481,  560)/
     . -81.495, -82.154, -82.170, -83.201, -83.886, -85.155, -84.177, -85.140,
     . -86.269, -85.980, -86.710, -87.682, -88.704, -88.412, -88.412, -89.516,
     . -90.747, -89.653, -88.833, -87.638, -87.299, -87.866, -88.649, -89.786,
     . -90.124, -90.124, -90.156, -91.283, -92.389, -93.413, -92.424, -91.424,
     . -90.677, -91.411, -92.356, -92.479, -68.183, -68.827, -69.286, -69.977,
     . -70.711, -69.996, -69.981, -70.505, -70.979, -71.299, -71.299, -70.650,
     . -70.129, -69.675, -69.193, -68.604, -67.955, -67.709, -84.100, -83.224,
     . -82.378, -82.318, -82.253, -82.205, -82.296, -81.986, -81.989, -81.542,
     . -80.973, -80.530, -80.887, -80.141, -79.543, -79.333, -79.332, -79.740,
     . -79.411, -78.944, -78.961, -78.570, -78.725, -78.936, -78.919, -78.920/
      data (Longitude(i),i=  561,  640)/
     . -79.062, -79.346, -79.761, -78.957, -78.500, -56.764, -57.552, -58.306,
     . -58.997, -59.463, -60.005, -60.783, -61.563, -61.590, -61.589, -62.359,
     . -63.148, -63.935, -64.719, -65.501, -66.276, -66.959, -67.229, -67.924,
     . -68.183, -67.709, -67.004, -66.299, -65.537, -64.781, -64.212, -64.539,
     . -65.126, -65.866, -66.596, -66.848, -66.849, -66.099, -65.476, -64.739,
     . -64.979, -64.875, -64.579, -63.895, -63.772, -63.772, -63.304, -62.591,
     . -61.931, -61.364, -61.650, -61.882, -61.884, -62.542, -63.200, -63.859,
     . -64.436, -64.884, -65.408, -65.757, -65.756, -66.161, -66.009, -65.454,
     . -64.934, -64.221, -63.567, -64.273, -64.536, -64.574, -92.470, -93.408,
     . -93.950, -94.166, -94.556, -94.764, -94.788, -94.803, -94.291, -93.357/
      data (Longitude(i),i=  641,  720)/
     . -93.166, -93.158, -92.955, -92.696, -92.426, -91.530, -90.609, -89.813,
     . -88.927, -88.247, -87.664, -86.865, -86.047, -85.338, -84.492, -84.098,
     . -64.574, -65.112, -65.725, -66.402, -67.112, -67.221, -67.877, -68.053,
     . -68.053, -68.754, -69.191, -69.807, -70.362, -70.730, -70.881, -70.623,
     . -70.064, -69.936, -69.941, -70.505, -71.164, -71.803, -72.467, -73.117,
     . -73.719, -74.208, -74.258, -74.248, -74.019, -74.299, -74.669, -75.299,
     . -75.413, -75.420, -75.191, -75.041, -75.041, -75.371, -75.602, -75.950,
     . -75.876, -75.820, -76.303, -76.214, -75.841, -76.481, -76.507, -76.453,
     . -77.076, -77.251, -77.246, -76.653, -76.310, -76.849, -76.427, -76.287,
     . -76.896, -76.509, -75.914, -75.851, -75.851, -75.674, -75.882, -76.195
      data (Longitude(i),i=  721,  800)/
     . -76.138, -76.139, -76.635, -76.759, -77.247, -77.437, -77.434, -77.732,
     . -78.062, -78.663, -79.093, -79.340, -79.342, -79.757, -80.261, -80.856,
     . -80.897, -81.154, -81.490, -81.489, -81.493, -81.392, -81.279, -81.082,
     . -80.836, -80.549, -80.524, -80.448, -80.447, -80.621, -80.487, -80.291,
     . -80.072, -80.060, -80.120, -80.343, -80.398, -80.399, -80.955, -81.229,
     . -81.621, -81.845, -81.777, -81.778, -82.242, -82.579, -82.408, -82.713,
     . -82.679, -83.045, -83.153, -83.154, -83.494, -83.923, -84.483, -85.014,
     . -85.504, -85.978, -86.453, -86.453, -87.019, -87.598, -88.007, -88.417,
     . -88.997, -89.528, -90.070, -90.304, -90.304, -89.756, -89.512, -89.149,
     . -89.410, -89.410, -89.778, -90.354, -90.899, -91.338, -91.338, -91.664/
      data (Longitude(i),i=  801,  880)/
     . -92.205, -92.778, -93.333, -93.885, -93.886, -94.414, -94.964, -95.054,
     . -95.457, -95.956, -96.526, -96.800, -96.800, -97.183, -97.292, -97.723,
     . -97.539, -97.373, -97.166, -97.165, -97.137,-117.122,-117.281,-117.282,
     .-117.613,-118.083,-118.554,-119.154,-119.646,-120.253,-120.447,-120.447,
     .-120.654,-120.844,-121.294,-121.644,-121.941,-121.745,-121.745,-122.221,
     .-122.488,-122.349,-121.711,-122.331,-122.949,-123.035,-123.034,-123.424,
     .-123.707,-123.764,-124.073,-124.339,-124.096,-124.069,-124.243,-124.242,
     .-124.404,-124.501,-124.371,-124.125,-124.097,-124.042,-123.997,-123.995,
     .-123.945,-123.952,-123.307,-123.992,-123.752,-123.751,-124.159,-124.348,
     .-124.672,-124.712,-124.713,-124.025,-123.279,-122.612,-123.056,-122.565/
      data (Longitude(i),i=  881,  960)/
     .-122.528,-122.330,-122.186,-128.535,-128.535,-128.881,-128.540,-129.071,
     .-128.881,-128.824,-128.824,-129.231,-129.231,-130.176,-129.592,-130.286,
     .-130.176,-130.391,-129.778,-130.498,-130.391,-130.272,-130.272,-130.336,
     .-130.336,-130.165,-130.165,-131.783,-131.828,-131.260,-131.943,-132.496,
     .-131.783,-131.663,-131.916,-132.166,-132.842,-133.115,-132.309,-132.600,
     .-132.044,-131.663,-139.313,-139.313,-135.936,-135.012,-134.671,-135.936,
     .-127.470,-128.242,-127.553,-127.907,-127.121,-126.603,-126.068,-125.621,
     .-124.869,-124.809,-124.808,-125.180,-124.532,-123.831,-123.583,-124.023,
     .-124.717,-125.127,-125.494,-126.256,-127.024,-127.477,-123.100,-123.100,
     .-122.896,-122.896,-123.577,-123.577,-123.675,-123.675,-124.609,-124.609/
      data (Longitude(i),i=  961, 1040)/
     .-124.983,-124.983,-125.245,-125.245,-125.273,-125.273,-126.252,-126.252,
     .-126.127,-126.127,-127.998,-127.998,-127.953,-127.953,-127.240,-127.699,
     .-127.271,-127.240,-127.962,-127.962,-128.148,-128.148,-128.373,-128.373,
     .-126.768,-126.768, -64.383, -64.383, -65.068, -65.068, -64.660, -64.660,
     . -70.588, -70.588, -73.412, -74.746, -73.412, -73.412, -75.010, -75.037,
     . -75.879, -77.147, -77.204, -76.469, -75.130, -75.010, -77.625, -77.625,
     . -76.571, -76.571, -77.501, -78.564, -77.496, -77.501, -82.186, -82.411,
     . -83.360, -83.559, -82.541, -82.186, -79.541, -79.454, -80.027, -79.541,
     . -78.661, -78.661, -79.127, -79.107, -79.973, -79.505, -79.127, -79.556,
     . -79.556, -78.227, -78.227, -74.481, -74.481, -67.949, -67.949, -64.640/
      data (Longitude(i),i= 1041, 1120)/
     . -64.640, -61.355, -61.355, -57.941, -57.941,-119.734,-118.123,-116.623,
     .-115.491,-116.614,-117.938,-119.186,-120.169,-120.395,-121.793,-122.892,
     .-123.701,-124.877,-125.516,-125.245,-125.264,-124.479,-124.396,-124.245,
     .-123.695,-121.849,-120.231,-119.745,-114.599,-113.321,-111.676,-111.677,
     .-110.085,-110.681,-109.112,-108.571,-108.232,-107.809,-107.806,-107.660,
     .-107.961,-108.274,-106.582,-105.446,-105.140,-104.801,-104.305,-104.414,
     .-104.414,-103.523,-102.218,-100.989,-102.293,-103.136,-101.857,-102.734,
     .-104.123,-104.266,-104.265,-105.546,-106.571,-107.257,-108.551,-109.743,
     .-110.009,-110.013,-111.379,-112.714,-113.618,-114.329,-115.757,-116.897,
     .-117.433,-117.431,-116.085,-114.617,-113.136,-111.645,-112.908,-114.378/
      data (Longitude(i),i= 1121, 1200)/
     .-115.871,-117.3[illegible] 118.415,-117.046,-116.816,-116.816,-115.364,-116.949,
     .-118.412,-118.709,-118.224,-116.993,-115.516,-114.048,-114.605, -83.015,
     . -82.042, -81.592, -80.532, -80.936, -82.012, -82.904, -83.872, -84.572,
     . -85.609, -85.710, -85.711, -86.769, -86.384, -86.380, -86.193, -86.080,
     . -85.014, -83.888, -83.022,-111.529,-111.529,-110.728,-110.728,-110.817,
     .-110.817,-109.883,-109.883,-109.577,-109.577,-109.447,-109.447,-108.951,
     .-108.951,-107.983,-107.983,-107.883,-107.883,-107.582,-107.582,-104.682,
     .-104.682,-101.694,-101.694,-100.086,-100.086,-100.324,-100.324,-100.096,
     .-100.096,-101.520,-101.520,-101.053,-101.053,-100.461,-100.461,-104.490,
     .-104.975,-106.343,-105.284,-104.490,-107.645,-107.645, -97.875, -96.917/
      data (Longitude(i),i= 1201, 1280)/
     . -96.234, -96.079, -97.447, -98.634, -98.500, -98.221, -97.875, -96.136,
     . -96.136, -95.489, -95.489, -86.435, -86.435, -83.577, -83.577, -82.936,
     . -82.936, -83.348, -83.348, -83.921, -83.921, -84.910, -84.910, -90.620,
     . -90.620, -90.720, -90.720, -55.617, -55.975, -56.100, -56.504, -56.840,
     . -56.238, -55.492, -55.491, -56.070, -55.348, -54.574, -53.817, -53.457,
     . -53.456, -53.978, -54.134, -53.438, -53.941, -53.830, -53.227, -52.917,
     . -52.920, -53.177, -52.808, -52.965, -53.637, -53.920, -54.250, -54.487,
     . -55.129, -55.386, -55.385, -55.269, -55.988, -56.721, -57.468, -58.209,
     . -58.429, -58.431, -59.170, -59.139, -58.604, -58.520, -57.859, -57.997,
     . -57.996, -57.713, -57.438, -56.978, -56.636, -55.937, -55.615, -81.088/
      data (Longitude(i),i= 1281, 1360)/
     . -80.671, -81.479, -81.088, -79.297, -79.297, -63.492, -62.766, -62.115,
     . -62.602, -63.511, -64.077, -63.492, -55.467, -55.467, -54.530, -54.530,
     . -54.081, -54.081, -53.564, -53.564, -54.125, -54.125, -56.289, -56.289,
     . -60.309, -60.509, -60.980, -60.322, -60.229, -60.939, -61.517, -61.181,
     . -60.834, -60.309, -61.408, -61.951, -61.408, -63.985, -63.865, -63.156,
     . -62.432, -62.510, -63.138, -63.833, -64.413, -63.985, -64.483, -64.483,
     . -60.095, -60.095,-133.102,-132.733,-133.197,-133.102,-133.586,-132.900,
     .-132.346,-131.969,-132.117,-132.557,-133.124,-133.245,-133.586,-133.297,
     .-133.297,-133.286,-133.286,-131.238,-131.238,-131.467,-131.467,-131.819,
     .-131.819,-131.232,-130.982,-131.663,-131.232,-132.390,-132.390,-132.336/
      data (Longitude(i),i= 1361, 1440)/
     .-132.336,-132.703,-132.703,-132.802,-132.802,-133.052,-133.126,-133.793,
     .-133.052,-134.087,-133.875,-134.406,-134.087,-135.105,-134.734,-134.622,
     .-135.319,-135.571,-135.105,-135.700,-135.700,-135.738,-134.912,-135.862,
     .-135.033,-135.668,-136.228,-135.857,-135.738,-136.454,-136.454,-134.872,
     .-134.132,-133.792,-134.232,-133.930,-134.376,-134.645,-134.754,-134.872,
     .-145.781,-145.781,-146.097,-146.097,-146.938,-147.477,-147.199,-146.938,
     .-152.342,-152.342,-152.091,-152.766,-152.140,-152.091,-153.261,-152.379,
     .-153.026,-153.608,-154.521,-154.653,-153.778,-153.915,-153.261,-152.895,
     .-152.895,-153.876,-153.876,-153.457,-153.457,-154.083,-154.083,-154.421,
     .-154.421,-155.557,-155.557,-159.872,-159.872,-159.516,-159.516,-160.696/
      data (Longitude(i),i= 1441, 1520)/
     .-160.696,-162.233,-162.233,-162.754,-162.754,-163.493,-164.085,-164.908,
     .-164.494,-163.644,-163.493,-160.710,-160.710, -74.250, -74.250, -75.321,
     . -75.321, -75.714, -75.482, -75.693, -75.714, -75.798, -75.798, -76.053,
     . -76.053, -76.225, -76.225, -76.703, -76.703, -77.359, -77.359, -77.949,
     . -77.949, -80.432, -80.244, -80.444, -80.432, -80.187, -80.187, -80.268,
     . -80.268, -80.699, -80.699, -80.789, -80.789, -80.940, -80.940, -81.052,
     . -81.052, -81.413, -81.413, -81.492, -81.492, -81.547, -81.547, -81.654,
     . -81.654, -81.757, -81.757, -82.184, -82.184, -84.709, -84.709, -86.559,
     . -87.135, -86.556, -86.559, -88.095, -88.095, -88.553, -88.553, -88.820,
     . -88.820, -89.236, -89.236, -91.724, -91.724, -94.813, -94.813, -96.244/
      data (Longitude(i),i= 1521, 1600)/
     . -96.244, -96.404, -96.404, -96.839, -96.839, -97.053, -97.325, -97.076,
     . -97.053, -97.380, -97.320, -97.188, -97.338, -97.392, -97.380, -82.115,
     . -82.115, -66.765, -66.765, -68.241, -68.241, -70.019, -70.019, -70.557,
     . -70.557, -71.912, -72.514, -73.159, -73.803, -73.319, -72.655, -71.998,
     . -71.912, -81.657, -82.305, -82.989, -82.307, -81.657, -83.337, -83.337,
     . -83.625, -83.625, -83.921, -83.921, -85.502, -85.502, -86.987, -87.288,
     . -87.073, -86.987, -87.974, -87.974, -88.471, -89.068, -88.497, -88.471,
     . -87.767, -87.767, -90.601, -90.601, -90.531, -90.531, -90.413, -90.413,
     .-118.530,-118.530,-118.517,-118.517,-119.493,-119.493,-119.868,-119.868,
     .-120.109,-120.109,-120.328,-120.328,-122.588,-122.588,-123.492,-122.755/
      data (Longitude(i),i= 1601, 1680)/
     .-122.105,-123.260,-124.506,-123.464,-122.251,-121.176,-119.959,-120.270,
     .-119.157,-118.079,-117.994,-118.512,-119.667,-119.936,-120.826,-120.357,
     .-121.554,-122.399,-123.492, -98.431, -98.553, -98.110, -97.326, -97.283,
     . -96.944, -96.959, -96.356, -96.446, -96.695, -96.692, -97.069, -97.232,
     . -97.454, -97.814, -98.159, -98.916, -99.152, -98.548, -98.452,-115.027,
     .-114.009,-113.445,-112.434,-111.735,-111.104,-110.075,-111.131,-110.126,
     .-109.094,-109.233,-109.233,-110.320,-111.333,-111.914,-112.804,-113.851,
     .-114.782,-115.747,-115.166,-115.061,-115.851,-116.811,-116.054,-115.022,
     . -86.229, -86.259, -86.534, -87.008, -87.620, -87.816, -87.841, -87.865,
     . -75.812, -76.311, -76.214, -76.754, -77.440, -78.115, -78.799, -79.058/
      data (Longitude(i),i= 1681, 1760)/
     . -79.058, -79.748, -79.338, -78.711, -78.027, -77.328, -76.713, -76.047,
     . -75.812, -81.655, -81.014, -80.343, -79.772, -79.088, -79.452, -79.731,
     . -82.423, -82.575, -82.760, -83.406, -83.408, -83.301, -83.427, -84.084,
     . -84.698, -84.990, -84.990, -85.313, -85.482, -86.180, -86.349, -86.456,
     . -86.357, -86.230, -87.866, -87.796, -87.696, -87.497, -87.847, -87.429,
     . -87.067, -86.355, -86.354, -85.674, -84.951, -84.229, -84.306, -85.036,
     . -85.713, -86.146, -86.145, -86.806, -87.485, -88.108, -88.727, -89.311,
     . -89.993, -90.412, -90.413, -90.930, -91.609, -91.353, -90.835, -90.153,
     . -89.579, -89.580, -89.205, -88.549, -87.874, -87.151, -86.389, -86.330,
     . -86.329, -86.064, -85.394, -84.935, -84.776, -84.592, -83.938, -83.598/
      data (Longitude(i),i= 1761, 1840)/
     . -83.599, -82.891, -82.168, -81.476, -80.754, -80.395, -80.005, -79.660,
     . -79.660, -80.232, -80.893, -81.316, -81.295, -81.606, -81.723, -81.885,
     . -82.415, -83.102, -82.440, -83.114, -83.175, -79.731, -80.324, -80.952,
     . -81.533, -82.184, -82.854, -83.432, -83.175, -83.102, -82.536, -81.986,
     . -81.655,-112.766,-112.458,-112.198,-112.751,-112.953,-112.766, -35.816,
     . -34.735, -33.933, -33.612, -32.970, -32.035, -32.427, -31.704, -30.362,
     . -29.110, -28.356, -28.359, -27.012, -25.816, -25.146, -23.988, -22.807,
     . -24.014, -25.394, -26.848, -28.214, -26.931, -28.335, -29.236, -29.234,
     . -28.008, -26.614, -25.411, -26.815, -28.395, -28.429, -27.209, -25.735,
     . -24.566, -24.100, -22.882, -21.606, -21.742, -21.742, -21.763, -22.606/
      data (Longitude(i),i= 1841, 1920)/
     . -23.866, -25.456, -24.662, -26.334, -25.086, -26.750, -25.404, -25.739,
     . -25.689, -25.691, -24.561, -22.786, -21.070, -21.154, -22.479, -20.694,
     . -20.719, -21.440, -22.472, -22.476, -20.745, -19.578, -21.406, -22.244,
     . -20.928, -18.735, -18.286, -20.539, -20.489, -21.729, -21.727, -21.639,
     . -21.135, -19.915, -19.791, -17.916, -20.462, -17.595, -19.532, -16.467,
     . -14.935, -14.941, -13.031, -14.471, -17.818, -20.926, -22.522, -23.956,
     . -23.141, -22.095, -25.526, -27.370, -30.751, -27.880, -24.176, -21.320,
     . -22.016, -22.018, -24.965, -29.136, -33.046, -29.158, -31.653, -36.134,
     . -38.143, -42.303, -43.999, -43.987, -41.496, -39.814, -42.204, -43.365,
     . -44.367, -47.045, -49.752, -49.918, -53.087, -54.004, -55.989, -55.983/
      data (Longitude(i),i= 1921, 2000)/
     . -59.225, -57.507, -60.092, -60.842, -63.364, -65.386, -67.272, -64.934,
     . -63.787, -63.778, -64.932, -65.710, -68.273, -70.230, -72.526, -72.342,
     . -70.087, -68.390, -66.661, -66.662, -68.309, -66.171, -68.447, -70.726,
     . -69.839, -69.082, -67.326, -65.319, -63.412, -61.498, -60.119, -60.116,
     . -58.424, -58.239, -56.944, -56.143, -55.773, -55.258, -55.096, -55.098,
     . -54.676, -55.635, -55.415, -55.520, -53.946, -53.485, -52.219, -51.346,
     . -51.346, -52.551, -51.071, -50.860, -52.321, -53.794, -53.163, -52.011,
     . -50.542, -50.837, -50.434, -51.044, -52.412, -52.500, -67.231, -68.314,
     . -67.846, -69.096, -67.725, -68.594, -70.018, -70.024, -70.023, -68.604,
     . -67.181, -67.686, -68.958, -70.384, -69.483, -68.284, -69.728, -71.078/
      data (Longitude(i),i= 2001, 2080)/
     . -70.519, -71.967, -72.623, -72.623, -71.984, -72.134, -73.180, -73.745,
     . -74.787, -74.549, -75.894, -75.799, -77.392, -77.805, -76.997, -76.997,
     . -78.533, -77.907, -78.985, -80.470, -80.755, -80.583, -81.209, -82.783,
     . -84.284, -83.659, -83.636, -83.634, -85.323, -84.007, -85.592, -85.515,
     . -84.370, -85.981, -84.959, -84.803, -86.292, -84.973, -85.889, -86.431,
     . -86.703, -86.405, -86.404, -85.664, -86.931, -88.394, -89.252, -89.746,
     . -89.682, -89.992, -88.894, -87.459, -87.375, -89.332, -88.336, -86.844,
     . -85.871, -84.394, -82.930, -81.470, -80.009, -78.800, -78.800, -79.287,
     . -77.843, -77.616, -76.188, -75.594, -76.537, -75.152, -74.221, -73.137,
     . -72.941, -88.357, -87.833, -88.279, -89.644, -89.920, -90.618, -90.437/
      data (Longitude(i),i= 2081, 2160)/
     . -91.865, -92.001, -91.514, -91.511, -92.189, -92.876, -93.399, -94.501,
     . -95.833, -96.377, -96.410, -96.409, -96.203, -96.532, -95.595, -94.369,
     . -94.387, -93.402, -94.277, -95.431, -99.996, -98.270, -97.167, -98.348,
     . -97.023, -96.615, -97.007, -98.181, -98.738, -98.739, -99.614,-100.514,
     .-101.889,-102.715,-101.081,-100.228,-101.581,-100.477, -99.998,-112.427,
     .-111.422,-109.588,-109.431,-109.707,-108.396,-107.003,-105.677,-106.792,
     .-107.772,-107.772,-109.494,-111.094,-112.794,-114.429,-112.911,-111.087,
     .-113.024,-114.906,-116.773,-115.737,-116.506,-115.475,-113.639,-112.433,
     . -92.005, -90.217, -91.071, -89.167, -87.568, -85.583, -84.144, -82.115,
. -80.256, -79.955, -79.331, -79.334, -80.753, -82.615, -84.496, -86.374/
 data (Longitude(i),i= 2161, 2240)/
. -88.264, -90.125, -90.814, -90.817, -92.384, -92.036, -92.734, -94.743,
. -96.403, -94.729, -93.260, -92.013, -76.248, -77.227, -73.447, -70.207,
. -66.391, -63.226, -61.219, -63.625, -66.686, -68.990, -70.024, -70.023,
. -67.208, -67.584, -69.281, -71.446, -73.745, -74.952, -77.667, -77.926,
. -77.923, -75.308, -77.987, -75.975, -75.234, -76.490, -77.890, -79.861,
. -81.738, -81.788, -81.783, -79.544, -77.872, -78.925, -80.638, -82.032,
. -84.181, -85.978, -86.629, -86.629, -88.616, -89.107, -87.129, -88.181,
. -85.856, -83.794, -82.723, -84.580, -85.530, -84.804, -86.688, -87.001,
. -84.683, -82.543, -82.245, -82.248, -84.065, -84.684, -86.442, -85.897,
. -83.267, -81.768, -82.658, -80.301, -78.025, -78.033, -78.552, -80.577/
 data (Longitude(i),i= 2241, 2320)/
. -83.697, -86.578, -85.075, -87.174, -85.907, -89.081, -88.406, -87.291,
. -87.288, -90.501, -90.610, -87.063, -85.502, -82.674, -79.578, -81.855,
. -81.285, -77.439, -76.261, -94.334, -91.414, -89.816, -87.963, -87.344,
. -85.020, -86.741, -87.969, -90.308, -92.790, -93.902, -91.830, -90.362,
. -90.371, -92.951, -95.676, -96.599, -94.209, -96.027, -94.943, -94.342,
. -87.354, -88.874, -89.331, -25.684, -25.900, -27.378, -26.568, -25.684,
. -23.112, -21.930, -23.588, -24.473, -23.112, -21.904, -23.613, -21.917,
. -21.904, -24.418, -22.944, -24.640, -24.418, -20.493, -20.493, -17.969,
. -17.969, -45.139, -47.163, -44.777, -45.139, -51.699, -51.699, -54.302,
. -52.996, -51.913, -52.783, -54.201, -54.872, -54.509, -54.302, -51.002/
 data (Longitude(i),i= 2321, 2400)/
. -51.002, -80.772, -79.038, -77.319, -76.353, -77.904, -79.526, -80.143,
. -80.798, -80.772, -77.114, -77.114, -78.459, -78.840, -78.471, -78.459,
. -78.142, -78.142, -80.479, -80.479, -95.088, -93.327, -91.571, -90.725,
. -91.477, -92.374, -94.058, -93.862, -95.161, -95.138, -95.653, -95.684,
. -95.326, -95.088, -97.654, -99.141, -97.654, -94.827, -93.493, -93.439,
. -95.344, -96.091, -94.827,-104.155,-104.155,-101.989,-101.989,-102.652,
.-102.652,-104.134,-104.134,-100.297,-100.297,-105.649,-104.463,-106.023,
.-105.649,-117.623,-118.222,-117.662,-117.623,-118.651,-116.488,-116.257,
.-116.794,-118.944,-119.491,-121.549,-121.422,-120.200,-118.894,-118.651,
.-114.190,-114.190,-109.792,-110.163,-112.057,-113.262,-111.364,-109.792/
 data (Longitude(i),i= 2401, 2480)/
.-114.284,-114.284,-110.394,-111.410,-110.431,-110.394,-101.672,-101.672,
.-103.703,-102.620,-100.143, -99.501,-101.969,-104.445,-103.412,-105.543,
.-103.703, -93.175, -95.204, -93.175, -97.951, -96.009, -94.948, -97.039,
. -97.868, -98.341, -97.951, -90.844, -90.844, -89.983, -89.983, -78.893,
. -78.893, -99.733, -98.792, -99.733, -75.224, -75.224,-100.384, -98.214,
. -97.518, -97.412, -97.640, -99.517,-100.531,-102.499,-101.478, -99.530,
.-100.691,-100.384, -20.231, -20.231, -18.923, -18.629, -19.056, -18.923,
. -17.675, -17.675, -19.314, -19.314, -19.194, -19.194, -17.721, -17.721,
. -19.086, -19.086, -72.236, -72.236, -71.239, -71.239, -53.438, -53.438,
. -53.584, -53.584, -52.971, -52.971, -90.195, -90.195, -90.692, -90.692/
 data (Longitude(i),i= 2481, 2560)/
. -91.476, -91.476, -96.772, -96.772, -21.192, -21.192, -18.323, -18.323,
. -18.152, -18.152, -54.971, -54.971, -35.818, -36.870, -37.427, -38.146,
. -39.025, -40.214, -40.217, -40.256, -40.566, -41.567, -40.622, -40.865,
. -41.749, -41.747, -41.538, -42.590, -42.242, -42.407, -43.079, -43.078,
. -42.694, -42.976, -43.990, -43.127, -44.096, -45.088, -44.486, -45.173,
. -44.774, -44.633, -44.635, -45.547, -45.466, -46.503, -47.519, -47.770,
. -47.769, -48.699, -48.935, -49.701, -50.072, -49.701, -49.701, -50.769,
. -51.382, -50.984, -50.176, -49.595, -49.586, -50.129, -50.902, -50.997,
. -51.737, -51.941, -52.069, -51.533, -50.551, -50.551, -51.771, -52.986,
. -52.041, -53.230, -52.700, -51.765, -50.566, -51.792, -52.701, -53.053/
 data (Longitude(i),i= 2561, 2640)/
. -52.647, -51.370, -50.350, -50.349, -51.579, -52.873, -52.078, -50.848,
. -52.035, -53.293, -52.287, -50.955, -50.770, -51.945, -52.491, -37.024,
. -37.024, -37.191, -37.191, -37.987, -37.987, -40.447, -40.447, -41.036,
. -41.036, -40.819, -40.819, -41.461, -41.461, -41.941, -41.941, -42.404,
. -42.404, -42.215, -42.215, -42.248, -42.248, -43.818, -43.818, -43.366,
. -43.366, -43.507, -43.507, -44.116, -44.116, -44.005, -44.005, -44.265,
. -44.265, -45.152, -45.152, -44.931, -44.931, -45.008, -45.008, -45.438,
     . -45.438, -45.357, -45.357, -46.277, -46.277, -46.538, -46.538, -47.005,
     . -47.005, -46.330, -46.330, -46.207, -46.207, -47.084, -47.084, -47.082,
     . -47.082, -47.270, -47.270, -47.758, -47.758, -53.188, -53.188, -97.137/
      data (Longitude(i),i= 2641, 2720)/
     . -97.269, -97.517, -97.644, -97.716, -97.747, -97.741, -97.846, -97.737,
     . -97.408, -97.753, -97.604, -97.358, -97.179, -97.179, -96.867, -96.531,
     . -96.325, -96.044, -95.662, -95.134, -94.709, -94.388, -94.388, -93.881,
     . -93.373, -92.846, -92.357, -91.859, -91.334, -91.135, -90.734, -90.729,
     . -90.729, -90.574, -90.481, -90.375, -89.987, -89.463, -88.928, -88.419,
     . -87.876, -87.350, -87.113, -87.112, -86.815, -86.982, -87.338, -87.440,
     . -87.639, -87.624, -87.773, -88.030, -88.283, -88.283, -88.136, -88.169,
     . -88.226, -88.348, -88.745, -88.911, -88.911, -88.394, -88.213, -88.213,
     . -87.730, -87.219, -86.697, -86.184, -85.671, -85.671, -85.149, -84.630,
     . -84.244, -84.244, -83.860, -83.389, -83.132, -83.132, -83.229, -83.378/
      data (Longitude(i),i= 2721, 2800)/
     . -83.518, -83.518, -83.533, -83.528, -83.678, -83.733, -83.733, -83.735,
     . -83.806, -83.644, -83.644, -83.450, -83.142, -82.799, -82.561, -77.889,
     . -78.211, -78.431, -77.951, -77.739, -77.739, -78.102, -78.572, -78.988,
     . -78.977, -78.977, -79.463, -79.531, -79.531, -79.603, -79.603, -79.919,
     . -80.389, -80.380, -80.381, -80.042, -80.405, -80.908, -81.035, -81.215,
     . -81.215, -81.705, -82.179, -82.432, -82.432, -82.854, -82.896, -82.561,
     . -82.229, -81.794, -81.567, -81.567, -81.061, -80.615, -80.132, -80.031,
     . -80.037, -79.828, -79.829, -79.357, -78.872, -78.371, -78.027, -78.027,
     . -77.679, -77.366, -82.897, -83.113, -83.619, -83.603, -83.603, -83.963,
     . -84.432, -84.721, -85.170, -85.285, -85.285, -84.923, -85.426, -85.818/
      data (Longitude(i),i= 2801, 2880)/
     . -85.673, -85.690, -85.690, -86.071, -86.465, -86.753, -87.165, -87.099,
     . -87.099, -87.465, -87.302, -87.302, -87.612, -87.817, -87.817, -88.269,
     . -88.778, -89.060, -89.060, -89.556, -90.024, -90.096, -90.096, -90.579,
     . -91.097, -91.586, -92.007, -92.247, -94.528, -95.031, -95.416, -95.912,
     . -96.418, -96.932, -97.434, -97.934, -98.423, -98.871, -99.372, -99.858,
     .-100.350,-100.650,-100.650,-101.118,-101.573,-101.950,-102.477,-102.983,
     .-103.483,-103.802,-104.258,-104.729,-105.109,-105.378,-105.378,-105.618,
     .-105.234,-105.223,-105.433,-105.655,-105.756,-106.116,-106.483,-106.814,
     .-107.180,-107.614,-108.015,-108.307,-108.776,-109.258,-109.319,-109.425,
     .-109.805,-109.807,-110.162,-110.581,-110.840,-111.326,-111.727,-112.081/
      data (Longitude(i),i= 2881, 2960)/
     .-112.402,-112.632,-112.832,-113.091,-113.186,-113.636,-113.636,-114.192,
     .-114.702,-114.873,-114.686,-114.644,-114.392,-113.954,-113.589,-113.183,
     .-112.872,-112.736,-112.336,-112.012,-111.955,-111.955,-111.799,-111.402,
     .-111.290,-110.981,-110.702,-110.695,-110.162,-109.814,-109.451,-109.661,
     .-110.154,-110.499,-110.917,-111.361,-111.813,-112.097,-112.063,-112.209,
     .-112.309,-112.309,-112.756,-113.210,-113.742,-114.198,-114.620,-115.049,
     .-114.485,-114.113,-114.154,-114.587,-114.955,-115.189,-115.187,-115.646,
     .-115.793,-116.036,-116.304,-116.647,-116.855,-117.122,-117.122, -94.529,
     . -94.032, -93.583, -93.180, -92.766, -92.786, -92.247, -92.611, -92.786,
     . -64.748, -64.748, -72.330, -72.330, -71.586, -71.586, -71.848, -71.848/
      data (Longitude(i),i= 2961, 3040)/
     . -72.005, -72.005, -73.145, -73.145, -73.065, -73.065, -73.706, -73.189,
     . -73.657, -73.706, -73.814, -73.814, -74.344, -74.344, -73.852, -74.122,
     . -73.852, -74.284, -74.284, -75.308, -75.138, -75.308, -74.834, -74.834,
     . -74.530, -74.530, -75.627, -75.524, -75.627, -75.651, -75.651, -76.020,
     . -76.020, -77.701, -77.547, -77.701, -78.205, -77.854, -77.985, -78.437,
     . -78.157, -78.205, -77.880, -77.880, -77.506, -77.506, -76.735, -76.285,
     . -76.178, -76.314, -76.735, -77.950, -77.403, -77.038, -77.193, -77.232,
     . -77.513, -77.950, -78.963, -78.399, -78.959, -78.963, -67.938, -67.938,
     . -67.259, -66.737, -66.208, -65.689, -66.008, -66.538, -67.066, -67.259,
     . -65.562, -65.562, -64.862, -64.862, -65.013, -65.013, -64.640, -64.640/
      data (Longitude(i),i= 3041, 3120)/
     . -64.407, -64.407, -64.732, -64.732, -63.168, -63.168, -63.135, -63.135,
     . -62.869, -62.869, -62.960, -62.960, -62.839, -62.839, -62.600, -62.600,
     . -61.868, -61.868, -61.826, -61.826, -62.202, -62.202, -61.522, -61.522,
     . -61.757, -61.757, -61.288, -61.288, -61.468, -61.468, -61.199, -60.826,
     . -61.228, -61.199, -71.989, -71.492, -70.959, -70.446, -69.928, -69.519,
     . -69.112, -68.615, -68.446, -68.973, -69.505, -70.033, -70.523, -71.039,
     . -71.265, -71.757, -72.247, -72.775, -73.298, -73.819, -74.332, -73.894,
     . -73.364, -72.840, -72.346, -72.729, -72.786, -73.293, -72.868, -72.358,
     . -71.989, -68.784, -68.784, -71.522, -71.522, -73.707, -73.707, -73.798,
     . -73.798, -73.295, -72.803, -73.299, -73.295, -72.948, -72.948, -83.926/
      data (Longitude(i),i= 3121, 3200)/
     . -83.926, -83.959, -83.959, -81.395, -81.395, -80.112, -80.112, -78.373,
     . -77.910, -77.382, -76.864, -76.370, -76.859, -77.366, -77.856, -78.341,
     . -78.373, -84.924, -84.397, -84.215, -83.756, -83.257, -82.714, -82.193,
     . -81.647, -81.123, -80.592, -80.099, -79.631, -79.192, -78.648, -78.161,
     . -77.747, -77.219, -76.834, -76.303, -75.779, -75.493, -74.956, -74.529,
     . -74.132, -74.132, -74.645, -75.153, -75.686, -76.215, -76.745, -77.281,
     . -77.112, -77.541, -78.076, -78.488, -78.633, -79.173, -79.692, -80.203,
     . -80.697, -81.186, -81.719, -81.874, -82.420, -82.946, -83.356, -83.898,
     . -84.328, -84.868, -84.925, -79.632, -79.632, -79.089, -79.089, -78.493,
     . -78.493, -78.311, -78.311, -78.015, -78.015, -77.947, -77.947, -78.461/
      data (Longitude(i),i= 3201, 3280)/
     . -78.461, -78.823, -78.823, -78.966, -78.966, -79.077, -79.077, -79.226,
     . -79.226, -79.384, -79.384, -81.562, -81.562, -83.075, -82.576, -83.080,
     . -83.075,-106.248,-106.248,-106.472,-106.472,-109.009,-109.009,-109.916,
     .-109.916,-110.394,-110.394,-110.701,-110.701,-111.142,-111.142,-112.438,
     .-112.438,-112.604,-112.604,-112.886,-112.886,-113.589,-113.193,-113.589,
     .-114.651,-114.651,-118.304,-118.304,-115.260,-115.260,-112.134,-112.167,
     .-112.140,-112.134,-112.011,-112.011,-110.816,-110.816,-110.998,-110.998,
     .-114.772,-114.772,-106.687,-106.687,-106.649,-106.649, -91.539, -91.539,
     . -86.745, -86.745, -87.255, -87.255, -87.328, -87.328,-109.218,-109.217,
     . -87.998, -87.997, -87.935, -87.935, -86.986, -86.987, -86.630, -86.628/
      data (Longitude(i),i= 3281, 3360)/
     . -85.952, -85.952, -87.647, -87.648, -87.588, -87.591, -82.800, -82.799,
     . -83.051, -83.050, -85.593, -85.589, -82.266, -82.263, -82.224, -82.222,
     . -82.194, -82.192, -79.105, -79.104, -79.091, -79.090, -78.872, -78.870,
     . -81.087, -81.084, -81.740, -81.738, -81.811, -81.810, -82.316, -82.316,
     . -82.274, -82.274, -82.329, -82.329, -97.437, -97.560, -97.679, -97.740,
     . -97.708, -97.657, -97.539, -97.437, -85.773, -85.333, -84.974, -85.338,
     . -85.773, -85.924, -85.771, -79.918, -79.912, -79.687, -79.562, -77.367,
     . -77.049, -76.720, -76.309, -76.259, -78.596, -78.561, -78.065, -77.669,
     . -77.408, -77.129, -77.429, -77.319, -77.372, -77.509, -77.396, -77.356,
     . -76.259, -75.953, -75.642, -75.546, -75.286, -74.913, -74.406, -74.046/
      data (Longitude(i),i= 3361, 3440)/
     . -74.029, -74.031, -73.528, -73.061, -72.619, -72.188, -71.827, -71.309,
     . -71.325, -77.356, -77.699, -77.890, -78.811, -79.174, -79.644, -80.105,
     . -80.040, -80.340, -80.489, -80.870, -80.813, -80.801, -80.800, -80.914,
     . -80.539, -80.055, -79.844, -80.103, -80.338, -80.337, -80.729, -81.054,
     . -81.319, -81.115, -80.959, -81.174, -80.840, -80.399, -79.975, -79.696,
     . -79.464, -79.158, -79.136, -79.136, -78.848, -78.666, -78.435, -78.248,
     . -78.044, -77.771, -77.664, -77.277, -77.134, -76.789, -76.520, -76.265,
     . -76.262, -76.271, -76.272, -76.142, -75.821, -75.419, -75.084, -74.621,
     . -74.160, -73.732, -73.257, -72.790, -72.306, -71.896, -71.434, -71.166,
     . -70.738, -70.404, -78.596, -79.018, -78.811, -70.404, -70.332, -70.267/
      data (Longitude(i),i= 3441, 3520)/
     . -70.147, -70.162, -70.161, -70.094, -70.146, -70.247, -70.298, -70.594,
     . -70.473, -70.493, -73.912, -73.867, -73.586, -73.019, -72.350, -72.815,
     . -72.478, -72.862, -72.908, -73.051, -73.282, -72.735, -72.589, -72.590,
     . -72.866, -73.336, -73.567, -73.635, -73.342, -73.342, -73.754, -74.412,
     . -74.926, -74.926, -75.536, -74.887, -74.159, -74.113, -74.686, -73.986,
     . -73.263, -74.008, -74.029, -74.055, -74.056, -74.401, -73.870, -73.884,
     . -74.654, -74.068, -73.803, -73.662, -73.465, -73.384, -73.387, -72.626,
     . -73.241, -72.509, -72.474, -73.292, -73.645, -73.215, -72.724, -71.942,
     . -71.475, -71.475, -72.286, -72.902, -72.537, -72.311, -71.540, -71.775,
     . -71.572, -70.981, -70.813, -70.201, -69.489, -68.679, -68.434, -70.493/
      data (Longitude(i),i= 3521, 3600)/
     . -70.541, -70.547, -70.447, -70.716, -70.673, -70.768, -70.952, -71.053,
     . -71.158, -71.339, -71.492, -71.317, -71.393, -71.711, -71.654, -71.570,
     . -71.565, -71.565, -71.528, -71.446, -71.680, -71.644, -71.926, -72.042,
     . -72.199, -72.496, -72.712, -72.878, -73.145, -73.626, -73.659, -73.467,
     . -73.469, -73.263, -73.347, -73.662, -73.783, -73.911, -68.440, -68.771,
     . -69.382, -69.144, -69.106, -68.579, -68.715, -68.192, -67.729, -67.656,
     . -67.226, -66.678, -66.104, -66.389, -66.389, -65.775, -65.945, -66.677,
     . -67.206, -67.586, -67.374, -67.043, -66.450, -65.746, -65.451, -65.237,
     . -65.255, -65.028, -65.028, -64.431, -65.005, -64.435, -63.947, -63.595,
     . -64.215, -64.887, -65.028, -65.109, -65.108, -65.067, -64.412, -63.827/
      data (Longitude(i),i= 3601, 3680)/
     . -63.158, -62.547, -62.394, -62.332, -62.069, -62.327, -61.689, -61.043,
     . -60.399, -59.765, -59.143, -58.533, -57.952, -57.536, -57.170, -56.842,
     . -56.674, -57.117, -57.387, -57.166, -57.599, -58.137, -58.484, -58.472,
     . -58.459, -58.137, -58.200, -58.200, -58.046, -58.387, -58.425, -58.042,
     . -57.495, -56.920, -56.381, -55.772, -55.176, -54.572, -54.049, -53.671,
     . -53.374, -53.374, -52.924, -52.566, -52.382, -52.089, -52.094, -51.641,
     . -51.366, -51.092, -50.702, -51.154, -51.441, -51.852, -51.336, -50.924,
     . -50.583, -50.292, -50.094, -49.841, -49.505, -49.086, -48.[illegible]5, -48.568,
     . -48.571, -48.629, -48.657, -48.556, -48.729, -48.188, -48.002, -38.920,
     . -38.954, -39.004, -39.031, -38.929, -39.009, -38.828, -38.324, -37.991/
      data (Longitude(i),i= 3681, 3760)/
     . -37.712, -37.510, -37.510, -37.344, -36.992, -36.604, -36.231, -35.891,
     . -35.553, -35.248, -35.150, -35.150, -34.978, -34.824, -34.812, -34.862,
     . -34.971, -35.104, -35.257, -35.581, -35.996, -35.996, -36.497, -36.981,
     . -37.409, -37.830, -38.196, -38.569, -38.978, -39.425, -39.859, -40.352,
     . -40.856, -41.084, -41.084, -41.589, -42.084, -42.564, -43.017, -43.517,
     . -4[illegible].998, -44.414, -44.749, -44.653, -44.375, -44.521, -44.938, -45.338,
     . -45.839, -46.001, -48.003, [illegible]85, -47.206, -46.808, -46.388, -45.851,
     . -45.331, -44.835, -44.727, -44.727, -44.415, -43.871, -43.340, -42.795,
     . -42.249, -41.944, -41.515, -41.016, -41.062, -40.964, -40.964, -40.702,
     . -40.336, -40.140, -39.794, -39.716, -39.731, -39.584, -39.214, -39.216/
      data (Longitude(i),i= 3761, 3840)/
     . -39.134, -39.014, -38.920, -46.001, -46.497, -46.943, -47.419, -47.915,
     . -48.298, -48.436, -48.894, -49.280, -49.459, -49.692, -49.692, -49.479,
     . -49.867, -50.356, -50.712, -50.711, -50.826, -51.329, -51.669, -51.669,
     . -52.128, -52.633, -52.193, -51.771, -51.709, -51.709, -51.553, -51.255,
     . -50.863, -50.513, -50.181, -49.901, -50.136, -50.582, -50.772, -50.935,
     . -51.066, -51.166, -51.467, -51.684, -51.684, -51.766, -52.180, -52.589,
     . -52.958, -53.451, -53.919, -54.168, -54.168, -54.128, -54.624, -55.126,
     . -55.633, -56.121, -56.618, -57.107, -57.248, -57.249, -57.146, -57.162,
     . -60.426, -59.926, -59.506, -59.060, -58.683, -58.472, -58.612, -58.123,
     . -57.712, -57.296, -57.158, -71.325, -71.798, -71.806, -71.584, -71.827/
      data (Longitude(i),i= 3841, 3920)/
     . -72.117, -71.831, -71.489, -71.061, -71.091, -71.378, -71.527, -71.133,
     . -70.655, -70.168, -70.295, -70.206, -70.206, -69.806, -69.406, -68.895,
     . -68.455, -68.303, -67.866, -67.361, -66.854, -66.342, -65.918, -65.441,
     . -64.930, -64.448, -63.974, -64.289, -64.289, -63.780, -63.271, -62.764,
     . -62.256, -62.747, -62.801, -62.439, -62.439, -61.941, -61.444, -60.989,
     . -61.099, -61.106, -61.602, -61.129, -61.106, -61.602, -61.110, -60.604,
     . -60.425, -81.352, -81.352, -81.688, -81.688, -79.914, -79.914, -67.096,
     . -67.093, -67.496, -67.498, -67.352, -67.354, -67.055, -67.917, -67.066,
     . -67.056, -68.396, -68.154, -68.952, -69.831, -69.093, -68.391, -69.203,
     . -69.198, -70.348, -70.351, -70.993, -70.992, -72.028, -72.028, -71.282/
      data (Longitude(i),i= 3921, 4000)/
     . -71.281, -70.484, -70.580, -70.574, -70.483, -68.642, -69.514, -70.385,
     . -71.216, -72.001, -71.136, -70.279, -70.887, -70.079, -69.228, -69.378,
     . -69.376, -69.653, -70.182, -69.479, -70.305, -70.115, -69.624, -68.799,
     . -68.616, -71.671, -71.671, -72.396, -72.398, -72.294, -72.292, -73.179,
     . -73.180, -72.185, -72.809, -73.268, -72.446, -72.186, -73.688, -73.689,
     . -73.430, -73.420, -74.056, -74.057, -74.671, -73.959, -73.267, -74.027,
     . -74.711, -74.664, -73.398, -73.395, -73.951, -73.952, -74.136, -74.147,
     . -73.717, -73.724, -74.239, -74.238, -74.393, -74.402, -74.017, -74.023,
     . -74.775, -74.778, -75.023, -75.021, -75.109, -75.120, -75.314, -75.315,
     . -75.017, -75.017, -74.428, -74.432, -74.889, -74.889, -74.223, -74.222/
      data (Longitude(i),i= 4001, 4080)/
     . -74.715, -74.714, -75.321, -75.[illegible]28, -75.317, -75.321, -74.781, -74.890,
     . -74.886, -74.497, -74.538, -74.784, -75.313,  75.312, -74.949, -74.958,
     . -75.523, -75.525, -74.885, -74.887, -75.274, -75.274, -75.634, -75.633,
     . -74.618, -74.762, -74.516, -74.623, -75.053, -75.057, -74.343, -74.342,
     . -75.599, -75.606, -75.242, -75.439, -75.196, -75.241, -75.247, -75.246,
     . -75.186, -75.181, -74.955, -74.959, -74.229, -74.230, -74.465, -74.464,
     . -75.063, -75.071, -74.700, -74.702, -74.486, -74.486, -73.931, -73.931,
     . -73.676, -73.677, -74.067, -74.069, -74.144, -74.142, -74.026, -74.030,
     . -73.790, -73.793, -74.388, -74.391, -74.333, -74.342, -74.306, -74.311,
     . -73.831, -73.844, -74.472, -74.473, -75.103, -75.100, -74.634, -74.635/
      data (Longitude(i),i= 4081, 4160)/
     . -74.803, -74.807, -74.285, -74.289, -73.771, -73.771, -73.303, -73.227,
     . -73.146, -73.301, -74.521, -74.523, -74.045, -74.049, -73.934, -73.967,
     . -74.399, -74.402, -74.039, -74.042, -73.867, -73.868, -73.993,  73.997,
     . -74.816, -74.816, -73.386, -73.392, -73.625, -73.628, -73.429, -73.438,
     . -73.894, -74.228, -74.202, -74.051, -73.440, -73.814, -73.520, -73.901,
     . -66.870, -66.864, -66.502, -66.499, -64.287, -64.287, -68.642, -68.644,
     . -68.638, -68.642, -68.615, -68.231, -68.000, -67.466, -66.764, -66.092,
     . -65.224, -65.965, -66.842, -67.714, -68.580, -68.638, -62.079, -62.079,
     . -48.567, -48.567, -48.591, -48.591, -50.831, -50.831, -50.674, -50.674,
     . -51.126, -51.126, -50.564, -50.564, -50.360, -50.360, -50.069, -50.069/
      data (Longitude(i),i= 4161, 4240)/
     . -49.896, -50.300, -49.894, -49.896, -49.428, -49.428, -48.396, -48.531,
     . -48.784, -49.191, -49.627, -50.131, -50.621, -50.775, -50.783, -50.591,
     . -50.091, -49.599, -49.105, -48.606, -48.396, -50.033, -50.033, -50.368,
     . -50.368, -50.471, -50.471, -51.586  -51.455, -51.360, -51.729, -51.686,
     . -32.398, -32.398, -45.260, -45.260, -44.331, -44.331, -70.050, -70.050,
     . -69.146, -68.749, -69.163, -69.146, -68.377, -68.377, -66.616, -66 616,
     . -66.193, -66.193, -65.353, -65.353, -64.609, -64.609, -63.976, -63.976,
     . -64.331, -63.820, -64.323, -64.331, -61.079, -60.972, -61.405, -61.915,
     . -61.461, -61.464, -61.079, -60.525, -60.525, -61.610, -61.610, -61.163,
     . -61.163, -60.938, -60.938, -59.505, -59.505, -45.214, -45.214, -44.731/
      data (Longitude(i),i= 4241, 4320)/
     . -44.731, -37.039, -37.039, -36.087, -36.651, -37.404, -36.565, -36.059,
     . -36.087, -27.584, -27.584, -27.163, -27.163, -26.661, -26.661, -26.426,
     . -26.426, -26.259, -26.259, -26.592, -26.592, -27.344, -27.344, -29.844,
     . -29.844, -30.308, -30.308, -29.361, -29.361, -29.366, -29.366, -61.029,
     . -60.311, -59.773, -59.850, -60.517, -61.030, -59.720, -59.221, -58.705,
     . -57.906, -58.300, -59.039, -59.715, -61.106, -61.107,-109.406,-109.406,
     .-105.463,-105.463, -80.757, -80.757, -78.987, -78.987, -79.882, -79.882,
     . -80.082, -80.082, -90.523, -90.523, -89.658, -89.658, -89.362, -89.362,
     . -90.261, -90.261, -90.479, -90.479, -90.800, -90.800, -91.495, -91.284,
     . -91.473, -90.988, -90.959, -91.205, -91.495, -91.666, -91.666, -90.793/
      data (Longitude(i),i= 4321, 4400)/
     . -90.793, -69.524, -69.135, -68.864, -69.379, -69.765, -69.944, -69.523,
     .  75.820,  76.884,  77.905,  78.214,  79.201,  80.424,  81.462,  82.327,
     .  83.602,  84.893,  86.170,  87.387,  88.125,  88.917, 110.733, 111.802,
     . 112.978, 114.044, 114.711, 115.937, 116.846, 117.981, 119.272, 120.539,
     . 121.719, 121.973, 136.242, 137.485, 138.676, 139.890, 141.139, 142.331,
     . 143.607, 144.568, 144.637, 162.059, 163.417, 164.897, 166.396, 167.800,
     . 168.703, 169.906, 170.794, 169.874, 169.867, 169.109, 169.187, 162.796,
     . 163.065, 163.506, 164.419, 164.916, 166.907, 164.395, 162.640, 160.547,
     .  81.462,  82.240,  82.321, -75.980, -74.236, -72.592, -70.967, -69.257,
     . -67.888, -66.937, -66.963, -67.501, -67.509, -67.829, -68.423, -68.575,
      data (Longitude(i),i= 4401, 4480)/
     . -68.257, -65.806, -64.931, -64.167, -63.946, -63.168, -63.172, -62.485,
     . -61.428, -60.978, -59.894, -58.889, -57.951, -56.968, -57.924, -58.697,
       -59.579, -60.533, -61.338, -62.095, -62.458, -61.228, -61.676, -62.872,
     . -63.561, -63.567, -64.144, -64.939, -65.503, -65.331, -65.487, -13.987,
     . -15.804, -16.331, -15.252, -14.202, -12.583, -11.505, -11.737, -10.611,
     . -10.106,  -8 901   -8.632,  -7.627,  -6.823,  -6.107,  -5.999,  -5.996
     .  -4.467,  -2.906,  -1.378,   0.194,   1.508,   2.853,   4.302,   5.804,
     .   7.170,   8.665,   9.989,   9.989,  11.097,  12.423,  13.748,  15.216,
     .  16.641,  17.998,  19.377,  20.734,  22.211,  23.716,  25.136,  26.538,
     .  27.979,  28.138,  28.141,  29.469,  30.914,  32.111,  33.011,  33.452,
      data (Longitude(i),i= 4481, 4560)
     .  34.817,  35.537,-167.004,-162.638,-156.730,-150.411,-146.292,-140.434,
     .-143.126,-148.133,-151.897,-153.221,-152.959,-152.298,-153.093,-154.883,
     .-155.337,-152.605,-149.649,-148.609,-145.584,-145.584,-110.090,-108.113,
     .-106.161,-104.316,-102.378,-100.491,-100.198,-101.489,-102.461,-100.796,
     . -99.180, -99.180,-100.672,-102.434,-103.541,-101.850,-100.124, -98.395,
     . -96.926, -95.188, -93.446, -91.714, -90.839, -90.830, -89.350, -87.641,
     . -85.967, -85.669, -83.929, -82.328, -81.301, -80.191, -78.870, -77.143,
     . -75.980, -68.255, -66.902, -67.366, -66.866, -66.470, -65.806, -65.488,
     . -64.770, -63.832, -63.566, -63.153, -62.517, -62.005, -62.222, -62.214,
     . -61.088, -61.712, -62.264, -60.799, -60.567, -59.848, -61.577, -60.689/
      data (Longitude(i),i= 4561, 4640)/
     . -61.552, -61.064, -61.061, -61.948, -63.373, -63.290, -64.838, -66.483,
     . -68.424, -69.988, -72.146, -74.282, -76.197, -77.430, -77.805, -77.803,
     . -78.477, -77.455, -76.273, -74.140, -73.654, -76.074, -78.424, -80.480,
     . -79.745, -77.643, -79.074, -81.564, -83.062, -83.281, -81.517, -80.576,
     . -80.580, -78.641, -76.169, -77.044, -79.892, -77.304, -75.182, -73.095,
     . -70.199, -67.469, -65.034, -62.399, -59.518, -59.100, -58.521, -58.526,
     . -56.409, -54.249, -51.063, -47.725, -44.187, -42.074, -39.365, -36.715,
     . -33.740, -31.114, -31.086, -28.503, -29.061, -28.658, -31.364, -34.056,
     . -36.099, -36.051, -34.892, -33.567, -31.653, -30.069, -28.578, -26.643,
     . -26.449, -26.446, -24.561, -22.515, -20.472, -18.483, -18.061, -17.223/
      data (Longitude(i),i= 4641, 4720)/
     . -15.865, -14.593, -13.981, -67.364, -66.935, -66.839, -67.597, -67.223,
     . -66.879,  69.690,  69.261,  67.989,  67.673,  69.145,  69.146,  68.237,
     .  67.790,  67.735,  69.328,  70.492,  71.309,  71.427,  72.306,  72.631,
     .  73.726,  74.985,  75.823,  88.917,  90.187,  91.404,  92.653,  93.915,
     .  95.170,  96.440,  97.706,  98.966, 100.091, 100.899, 102.071, 103.304,
     . 104.112, 104.119, 105.298, 106.453, 107.706, 108.688, 109.960, 110.721,
     . 110.733, 121.973, 123.171, 124.289, 125.524, 126.790, 127.243, 128.469,
     . 129.379, 130.116, 131.354, 132.588, 133.827, 134.113, 134.176, 135.182,
     . 135.029, 135.912, 136.242, 144.637, 145.477, 146.711, 148.049, 148.668,
     . 149.999, 151.012, 152.409, 153.732, 153.633, 154.559, 155.532, 156.950/
      data (Longitude(i),i= 4721, 4800)/
     . 158.318, 159.436, 160.454, 161.724, 162.059, 169.187, 167.438, 166.033,
     . 164.767, 165.333, 163.750, 162.625, 162.927, 162.796, 160.551, 160.467,
     . 159.109, 158.400, 161.200, 160.741, 161.997, 163.766, 165.240, 167.090,
     . 169.917, 172.596, 176.784,-179.982,-174.942,-174.102,  35.537,  36.289,
     .  37.561,  38.063,  39.449,  39.812,  40.536,  41.678,  42.745,  44.013,
     .  45.143,  46.292,  47.436,  48.607,  48.642,  49.918,  50.858,  50.860,
     .  50.169,  50.986,  51.903,  53.106,  54.338,  55.537,  56.301,  57.339,
     .  58.206,  59.130,  60.436,  61.684,  63.000,  64.322,  65.622,  66.938,
     .  68.257,  69.538,  69.775,  69.718,  69.692,-174.097,-169.169,-167.007,
     .-148.609,-147.243,-145.523,-146.363,-148.448,-150.827,-153.103,-155.216/
      data (Longitude(i),i= 4801, 4880)/
     .-153.866,-155.955,-158.103,-158.083,-158.083,-157.806,-155.593,-153.428,
     .-151.258,-149.662,-147.816,-145.982,-146.135,-148.164,-146.051,-146.051,
     .-144.243,-142.629,-142.631,-140.622,-139.349,-137.428,-135.882,-133.961,
     .-132.045,-130.142,-128.290,-127.006,-127.006,-125.113,-123.225,-121.318,
     .-119.461,-117.919,-116.167,-111.794,-113.188,-112.103,-111.547,-110.078,
     .-110.959,-110.089,-148.001,-148.001, 167.108, 169.286, 167.207, 166.682,
     . 167.108, 169.897, 169.897, 164.852, 164.852, 163.118, 163.118, 162.500,
     . 162.500,  92.403,  92.403, -90.808, -91.143, -91.619, -90.808, -74.221,
     . -74.351, -75.966, -75.386, -74.221, -73.502, -73.502, -67.742, -67.742,
     . -69.221, -68.654, -67.966, -67.673, -68.401, -69.221, -66.800, -66.800
      data (Longitude(i),i= 4881, 4960)
     . -66.162, -66.162, -63.542, -63.542, -64.254, -63.532, -63.590, -64.254,
     . -62.617, -62.617, -61.141, -60.118, -61.136, -61.141, -59.897, -59.897,
     . -59.674, -59.674, -59.200, -59.200, -58.976, -58.071, -58.944, -58.976,
     . -55.343, -55.343, -54.133, -54.133, -56.318, -56.318, -56.416, -55.526,
     . -56.413, -56.416, -56.000, -56.000, -57.956, -58.285, -57.167, -57.956,
     . -57.475, -57.475, -59.517, -59.517,  -5.996,  -5.996,  -2.719,  -2.719,
     .  -2.762,  -2.762,   2.923,   2.923,  13.145,  13.145,  16.134,  16.134,
     .-100.851, -99.187,-100.850,-100.851,-102.466,-101.030, -99.450, -97.916,
     . -96.566, -97.784, -99.443,-100.981,-102.466, -90.379, -90.379, -78.326,
     . -78.326, -79.191, -79.085, -77.559, -78.802, -79.191, -76.519, -75.341/
      data (Longitude(i),i= 4961, 5040)/
     . -74.006, -74.799, -76.299, -76.519, -75.587, -75.587, -72.677, -72.677,
     . -72.491, -71.087, -71.620, -73.237, -74.816, -75.142, -73.550, -73.066,
     . -71.497, -71.985, -71.392, -71.634, -72.166, -71.279, -69.994, -69.472,
     . -69.081, -68.503, -68.278, -68.164, -68.427, -69.171, -70.749, -70.417,
     . -72.491, -57.376, -57.376, -60.934, -60.934, -62.174, -62.360, -61.976,
     . -62.174, -61.076, -61.076, -60.722, -60.722, -71.765, -71.392, -69.576,
     . -67.391, -68.516, -69.791, -71.765, -62.441, -64.417, -67.031, -64.094,
     . -61.714, -60.334, -62.441, -52.938, -54.551, -52.426, -51.101, -50.677,
     . -50.258, -49.166, -47.303, -44.988, -43.899, -43.799, -43.360, -43.625,
     . -45.216, -48.047, -51.036, -52.938, -37.802, -40.346, -37.797, -37.802/
      data (Longitude(i),i= 5041, 5120)/
     . -20.294, -21.565, -20.073, -20.294,-105.176,-105.176, -60.665, -60.665,
     . -61.474, -61.474, -60.452, -60.452,  71.831,  71.831,  96.543,  96.543,
     . 100.496, 100.496, 103.318, 103.318,  48.321,  48.321,  47.408,  47.408,
     .-159.152,-161.832,-163.654,-163.031,-160.590,-159.009,-159.152,-149.220,
     .-149.220,-132.372,-132.372,-130.896,-130.896,-127.118,-127.118,-124.092,
     .-125.676,-127.163,-125.809,-124.441,-123.730,-124.092,-118.651,-120.269,
     .-122.076,-122.406,-120.630,-118.814,-118.651,-115.989,-115.989,  28.833,
     .  28.636,  28.583,  28.582,  28.110,  27.900,  27.485,  27.909,  28.013,
     .  28.013,  28.243,  28.840,  28.227,  27.572,  27.128,  26.587,  26.064,
     .  26.032,  26.032,  25.408,  24.749,  24.111,  24.297,  23.639,  23.000/
      data (Longitude(i),i= 5121, 5200)/
     .  22.721,  22.729,  22.729,  23.109,  23.011,  23.330,  23.884,  24.064,
     .  24.064,  23.475,  23.314,  22.899,  23.041,  22.492,  22.489,  22.489,
     .  22.263,  21.657,  21.618,  21.158,  21.637,  22.278,  22.847,  22.507,
     .  21.867,  21.230,  21.176,  21.176,  20.891,  20.549,  20.172,  20.009,
     .  20.010,  19.679,  19.400,  19.452,  19.444,  19.368,  19.369,  18.911,
     .  18.345,  17.805,  17.157,  16.807,  16.133,  15.599,  15.154,  14.893,
     .  14.682,  14.051,  13.570,  13.717,  13.717,  13.001,  12.314,  12.460,
     .  12.278,  12.573,  12.575,  13.132,  13.655,  13.870,  14.157,  14.661,
     .  15.228,  15.904,  15.988,  16.580,  17.197,  17.747,  17.954,  17.954,
     .  18.420,  18.354,  17.928,  17.319,  16.708,  16.522,  17.072,  17.139/
      data (Longitude(i),i= 5201, 5280)/
     .  16.548,  16.301,  16.000,  16.000,  15.865,  16.218,  16.041,  15.790,
     .  15.261,  14.909,  14.909,  14.488,  13.992,  13.448,  12.818,  12.286,
     .  11.800,  11.777,  11.777,  11.265,  10.792,  10.524,  10.277,   9.846,
     .   9.229,   8.527,   8.118,   7.529,   7.529,   6.951,   6.441,   5.756,
     .   5.123,   4.451,   3.761,   3.214,   3.053,   3.210,   3.210,   3.254,
     .   2.714,   2.176,   1.538,   0.926,   0.553,   0.449,   0.449,   0.043,
     .  -0.281,  -0.220,   0.065,  -0.466,  -0.733,  -0.721,  -0.721,  -1.351,
     .  -1.814,  -2.098,  -2.716,  -3.338,  -3.962,  -4.458,  -4.457,  -5.027,
     .  -5.465,  -6.074,  -6.398,  -6.707,  -7.321,  -7.414,  -7.414,  -7.993,
     .  -8.602,  -8.828,  -8.824,  -9.198,  -8.970,  -9.493,  -9.493,  -9.358/
      data (Longitude(i),i= 5281, 5360)/
     .  -9.086,  -8.787,  -8.765,  -8.668,  -8.819,  -8.750,  -8.750,  -8.886,
     .  -9.137,  -8.689,  -8.161,  -7.908,  -7.908,  -7.262,  -6.572,  -5.884,
     .  -5.218,  -4.541,  -4.483,  -4.483,  -3.808,  -3.143,  -2.451,  -1.769,
     .  -1.770,  -1.408,  -1.282,  -1.238,  -1.152,  -0.595,  -0.805,  -1.244,
     .  -1.244,  -1.192,  -1.773,  -2.035,  -2.010,  -2.010,  -2.539,  -3.183,
     .  -3.868,  -4.564,  -4.727,  -4.727,  -4.745,  -4.030,  -3.289,  -2.701,
     .  -2.314,  -2.315,  -1.560,  -1.572,  -1.841,  -1.181,  -0.411,  -0.218,
     .  -0.218,   0.484,   0.905,   1.521,   1.567,   2.183,   2.542,   2.542,
     .   3.233,   4.030,   4.040,   4.021,   4.021,   4.532,   4.713,   5.412,
     .   5.915,   4.104,   3.827,   4.104,   4.046,   4.168,   3.873,  10.903
      data (Longitude(i) i= 5361, 5440)
     .  10.479,   9.928,   8.805,   8.472,   8.133,   8.106,   8.137,  10.340,
     .  10.748,  10.023,   9.608,   9.621,   8.137,   8.957    9.760,   8.847,
     .   8.586,   8.586,   9.496,   9.940,   9.940,  10.542,  10.542,   9.981,
     .   9.980,  10.339,   9.621,   9.749,   9.928,  10.610,  10.781,  10.758,
     .  12.443,  12.600,  11.714,  11.741,  12.557,  12.966,  13.647,  13.782,
     .   8.805,   8.815,   9.389,   8.543,   7.709,   7.016,   6.172,   5.915,
     .   8.959,   8.982,  14.226,  13.712,  13.712,  12.967,  12.175,  11.480,
     .  10.903,  19.624,  19.005,  19.004,  18.431,  17.566,  16.770,  16.103,
     .  15.271,  14.483,  14.226,  24.163,  25.211,  25.376,  24.524,  23.925,
     .  23.218,  22.453,  22.401,  22.401,  21.557,  21.078,  21.334,  21.538/
      data (Longitude(i),i= 5441, 5520)/
     .  21.450,  21.300,  21.300,  21.453,  22.408,  23.137,  23.225,  27.860,
     .  26.867,  25.858,  24.968,  24.015,  23.083,  23.226,  11.427,  11.248,
     .  11.794,  11.909,  12.236,  12.677,  12.557,  13.782,  14.237,  14.952,
     .  15.837,  16.211,  16.445,  16.649,  16.702,  16.701,  16.801,  16.194,
     .  17.139,  17.666,  18.602,  17.614,  16.703,  17.685,  18.646,  18.816,
     .  18.111,  17.232,  17.174,  17.274,  17.273,  17.403,  17.466,  17.954,
     .  18.591,  19.494,  20.033,  20.033,  20.852,  21.469,  21.045,  21.266,
     .  22.347,  23.430,  24.162,  30.784,  29.457,  30.733,  30.087,  28.571,
     .  28.197,  27.904,  26.773,  26.681,  25.817,  25.158,  25.281,  23.809,
     .  23.476,  22.261,  21.208,  21.209,  22.045,  20.592,  20.024,  20.382/
      data (Longitude(i),i= 5521, 5600)/
     .  18.995,  18.004,  17.470,  16.492,  17.838,  16.479,  16.252,  14.918,
     .  14.786,  14.786,  15.589,  14.337,  15.631,  14.351,  13.565,  13.216,
     .  12.607,  12.687,  11.800,  11.523,  10.512,  10.512,   9.976,  11.048,
     .  10.267,   9.162,   8.298,   7.201,   8.115,   7.054,   6.991,   5.977,
     .   5.086,   5.086,   5.921,   4.941,   5.220,   6.249,   7.292,   6.281,
     .   5.245,   5.261,   5.703,   6.459,   5.954,   5.521,   6.514,   6.034,
     .   5.457,   5.456,   6.073,   6.907,   7.852,   8.636,   9.278,  10.082,
     .  10.304,  11.007,  11.428,   5.376,   5.047,  25.657,  25.657,  24.655,
     .  24.655,  25.442,  25.442,  23.611,  23.611,  24.490,  24.490,  23.300,
     .  23.782,  24.214,  23.593,  23.183,  23.300,  24.374,  24.374,  24.441/
      data (Longitude(i),i= 5601, 5680)/
     .  24.441,  24.794,  24.794,  25.054,  25.054,  24.930,  24.930,  25.349,
     .  25.349,  25.572,  25.572,  25.308,  25.308,  25.384,  25.384,  26.321,
     .  25.834,  25.224,  24.634,  24.034,  23.570,  24.181,  24.761,  25.368,
     .  25.954,  26.321,  22.957,  22.957,  24.544,  24.544,  25.278,  25.278,
     .  24.653,  24.653,  23.501,  23.501,  23.547,  23.547,  20.706,  20.706,
     .  20.571,  20.571,  20.701,  20.701,  19.855,  19.855,  17.412,  17.412,
     .  16.861,  16.861,  16.721,  16.721,  16.388,  17.071,  16.384,  16.388,
     .  16.452,  16.452,  16.069,  16.069,  14.866,  14.866,  14.745,  15.245,
     .  14.735,  14.745,  14.358,  14.501,  14.301,  14.358,  14.594,  14.594,
     .  13.118,  13.118,  11.006,  11.006,   8.281,   8.281,   7.622,   7.622/
      data (Longitude(i),i= 5681, 5760)/
     .   7.339,   7.339,   6.797,   6.797,   6.328,   6.328,  12.055,  12.055,
     .  15.624,  15.280,  15.091,  15.120,  14.500,  14.023,  13.441,  12.911,
     .  12.485,  13.068,  13.676,  14.312,  14.909,  15.536,  15.624,  10.441,
     .  10.441,   9.256,   9.679,   9.711,   9.691,   9.605,   9.019,   8.419,
     .   8.383,   8.393,   8.376,   8.186,   8.849,   9.256,  14.324,  14.324,
     .  14.507,  14.507, -25.193, -25.193, -25.843, -25.241, -25.843, -27.380,
     . -27.380, -28.549, -28.549, -31.201, -31.201,   4.224,   4.224,   3.186,
     .   3.295,   2.656,   3.093,   3.186,   1.558,   1.558,   1.449,   1.449,
     .  -0.748,  -0.748,  -7.871,  -7.871,   9.430,   9.527,   9.458,   9.281,
     .   8.672,   8.546,   8.960,   9.415,   9.430,  -1.229,  -1.229,  -1.299/
      data (Longitude(i),i= 5761, 5840)/
     .  -1.299,  -2.370,  -2.370,  -2.158,  -2.158,  -3.236,  -3.236,  -5.049,
     .  -5.049,  -2.082,  -2.082,  -2.500,  -2.500,   5.836,   5.836,   4.886,
     .   4.886,   5.580,   5.580,   5.674,   5.674,  -7.266,  -6.411,  -5.816,
     .  -5.455,  -6.029,  -6.077,  -6.103,  -9.628,  -9.329, -10.007,  -9.607,
       -10.109,  -9.259,  -8.455,  -8.434,  -7.707,  -7.266,  -9.960,  -9.960,
     .  -3.979,  -3.191,  -2.253,  -2.066,  -2.438,  -3.144,  -3.325,  -3.325,
     .  -2.620,  -1.937,  -1.568,  -1.303,  -1.184,  -1.185,  -0.472,  -0.172,
     .  -0.260,  -0.261,   0.291,   0.215,   1.011,   1.714,   1.629,   1.157,
     .   0.487,   1.266,   1.442,   1.442,   0.971,   0.248,  -0.541,  -1.328,
     .  -2.079,  -2.870,  -3.530,  -4.305,  -5.010,  -5.072,  -4.474,  -3.759
      data (Longitude(i),i= 5841, 5920)
     .  -2.976,  -2.451,  -3.135,  -3.911,  -4.719,  -4.518,  -4.000,  -4.697,
     .  -4.760,  -4.760,  -4.218,  -3.421,  -2.898,  -2.790,  -3.559,  -3.298,
     .  -4.044,  -4.044,  -4.903,  -4.805,  -4.888,  -5.511,  -5.614,  -5.216,
     .  -5.122,  -5.122,  -5.909,  -5.789,  -5.830,  -5.194,  -5.263,  -5.263,
     .  -5.029,  -4.086,  -3.131,  -3.497,  -3.979,  -6.103,  -6.041,  -6.245,
     .  -6.401,  -6.402,  -7.152,  -7.873,  -8.541,  -9.303, -10.088, -10.339,
     . -10.338,  -9.771,  -9.369,  -9.628,  -6.244,  -6.383,  -6.272,  -6.244,
     .  -7.185,  -7.185,  -7.344,  -7.344,  -7.350,  -7.350,  -7.435,  -7.435,
     .  -6.323,  -5.905,  -6.773,  -6.323,  -6.326,  -6.326,  -6.487,  -6.487,
     .  -6.821,  -6.821,  -6.134,  -6.134,  -5.696,  -5.696,  -6.119,  -6.119/
      data (Longitude(i),i= 5921, 6000)/
     .  -6.117,  -6.117,   0.798,   0.798,  -5.072,  -4.775,  -4.474,  -3.042,
     .  -3.042,  -3.194,  -3.194,  -1.074,  -1.074,  -6.295,  -6.295,  -4.416,
     .  -4.416,  -4.353,  -4.353,  -5.266,  -5.266,  -8.552,  -8.552,  -0.795,
     .  -0.795,  -1.191,  -1.347,  -1.191,  -0.997,  -0.997,  11.231,  11.231,
     .  12.443,  12.120,  11.558,  11.099,  11.753,  11.741,  10.758,   9.908,
     .  10.614,  10.610,  10.557,  10.557,   9.076,   9.076,  14.776,  14.776,
     .  12.281,  12.281,  10.959,  10.959,   9.754,   9.754,  10.927,  10.927,
     .   8.928,   8.928,  25.552,  25.552,  23.749,  23.749,  23.156,  23.156,
     .  22.138,  23.340,  22.175,  22.138,  22.976,  22.976,  19.084,  19.084,
     .  18.191,  18.191,  20.468,  20.468,  19.932,  19.932,  19.902,  19.902/
      data (Longitude(i),i= 6001, 6080)/
     .  17.041,  17.344,  17.103,  17.041,  15.420,  15.420,  14.985,  15.538,
     .  15.046,  14.985,  14.811,  14.811,  14.211,  14.211,  13.489,  13.489,
     .  13.144,  13.144,  12.447,  12.447,  11.099,  11.099,   8.827,   8.827,
     .   9.144,   9.144,   8.184,   8.184,   6.039,   6.039,   5.803,   5.803,
     .   5.222,   5.222,   5.689,   5.689,   5.345,   5.345,   5.205,   5.205,
     .  11.737,  11.737,  17.110,  16.851,  16.570,  16.629,  17.010,  17.110,
     .  19.004,  18.927,  18.336,  18.110,  18.676,  19.004,  19.313,  19.313,
     .  19.225,  19.225,  24.586,  24.586,  21.418,  21.418,  22.954,  22.954,
     .  19.906,  19.906,  21.656,  21.656,  21.856,  21.856,  22.446,  22.446,
     . -14.559, -14.653, -13.805, -13.605, -14.417, -14.746, -14.746, -15.804/
      data (Longitude(i),i= 6081, 6160)/
     . -16.667, -17.777, -18.612, -19.709, -20.714, -21.365, -21.365, -22.502,
     . -21.705, -22.361, -23.454, -24.060, -24.058, -22.896, -21.755, -22.503,
     . -23.694, -23.677, -22.454, -23.139, -21.927, -21.377, -21.380, -21.446,
     . -20.325, -19.429, -18.239, -17.256, -16.270, -15.137, -14.558,  -6.717,
     .  -6.717,  -6.744,  -6.744,  -6.887,  -6.887,  -6.542,  -6.542,  -7.173,
     .  -7.173,  -7.409,  -7.409,  19.247,  19.894,  19.262,  19.247,   5.044,
     .   5.082,   5.871,   5.412,   5.376,  32.305,  31.882,  31.882,  31.315,
     .  30.788,  30.788,  30.228,  30.290,  30.290,  29.728,  29.225,  29.029,
     .  33.554,  33.850,  34.010,  34.255,  34.544,  34.778,  34.999,  35.278,
     .  35.661,  35.498,  35.663,  36.044,  36.446,  36.568,  29.028.  28.503/
      data (Longitude(i),i= 6161, 6240)/
     .  27.922,  27.374,  26.815,  26.241,  25.649,  25.152,  33.555,  33.422,
     .  33.047,  32.752,  32.752,  32.603,  32.410,  32.580,  25.152,  24.753,
     .  24.161,  23.609,  23.093,  22.643,  22.075,  21.488,  20.909,  20.403,
     .  20.005,  19.953,  20.143,  20.143,  19.954,  19.530,  18.966,  18.478,
     .  18.002,  17.461,  16.901   16.315,  15.757,  15.429,  15.289,  14.724,
     .  14.205,  13.613,  13.017,  12.421,  11.889,  11.528,  11.528,  11.113,
     .  10.524,  10.059,  10.343,  10.784,  11.094,  11.098,  11.098,  10.820,
     .  10.456,  10.828,  11.072,  10.503,  10.208,   9.588,   8.999,   8.622,
     .   8.622,   8.004,   7.488,   7.487,   6.894,   6.289,   5.716,   5.107,
     .   4.527,   3.899,   3.293,   2.693,   2.308,   2.308,   1.692,   1.074/
      data (Longitude(i),i= 6241, 6320)/
     .   0.517,   0.063,  -0.529,  -1.117,  -1.558,  -2.139,  -2.199,  -2.199,
     .  -2.806,  -3.424,  -4.037,  -4.654,  -5.148,  -5.621,  -5.929,  -5.928,
     .  -6.114,  -6.318,  -6.590,  -6.917,  -7.430,  -7.971,  -8.519,  -8.881,
     .  -9.251,  -9.352,  -9.352,  -9.689,  -9.838,  -9.888,  -9.639,  -9.908,
     . -10.221, -10.616, -11.104, -11.485, -12.010, -12.558, -13.055, -13.176,
     . -15.678, -15.935, -16.197, -16.346, -16.714, -16.960, -17.029, -17.092,
     . -17.056, -13.176, -13.394, -13.578, -14.037, -14.489, -14.664, -14.845,
     . -14.896, -15.289, -15.678, -17.055, -16.553, -16.201, -16.288, -16.361,
     . -16.149, -16.051, -16.041, -16.172, -16.388, -16.528, -16.528, -16.528,
     . -16.661, -16.949, -17.336, -17.005, -16.782, -16.548, -16.037, -15.525
      data (Longitude(i),i= 6321, 6400)/
     . -16.027, -16.529, -16.783, -16.364, -15.851, -15.390, -15.815, -16.328,
     . -16.720, -16.720, -16.206, -15.779, -15.272, -15.477, -15.254, -14.746,
     . -14.664, -14.276, -13.852, -13.518, -13.306, -13.302, -13.298, -13.181,
     . -12.924, -12.642, -12.306, -11.833, -11.419, -11.012, -10.640, -10.189,
     .  -9.816,  -9.456,  -9.053,  -8.604,  -8.148,  -7.682,  -7.526,  -7.526,
     .  -7.057,  -6.596,  -6.124,  -5.639,  -5.134,  -4.630,  -4.129,  -4.634,
     .  -4.137,  -3.652,  -3.182,  -3.104,  -3.104,  -2.611,  -2.147,  -1.676,
     .  -1.198,  -0.721,  -0.271,   0.185,   0.685,   1.115,   1.199,   1.199,
     .   1.688,   2.191,   2.695,   3.202,   3.672,   4.141,   4.617,   4.981,
     .   5.432,   5.492,   5.583,   5.995,   6.500,   6.753,   7.160,   7.664/
      data (Longitude(i),i= 6401, 6480)/
     .   7.764,   7.764,   8.270,   8.775,   8.988,   8.988,   9.385,   9.810,
     .   9.942,   9.844,   9.815,   9.815,   9.752,   9.486,   9.849,   9.804,
     .   9.804,   9.622,   9.985,   9.492,   9.298,   9.113,   9.113,   8.982,
     .   9.288,   9.560,   9.587,  10.045,  10.292,  10.642,  10.981,  11.15[illegible],
     .   9.557,   9.107,   9.317,   9.580,   9.587,  11.160,  11.512,  11.811,
     .  12.011,  13.182,  12.685,  12.216,  12.162,  12.011,  12.989,  12.735,
     .  12.518,  12.353,  12.282,  12.106,  11.997,  11.787,  11.819,  11.762,
     .  11.755,  13.182,  12.695,  12.446,  12.7[illegible]6,  12.912,  13.108,  13.322,
     .  13.367,  12.990,  13.145,  13.145,  13.284,  13.504,  13.773,  13.829,
     .  13.786,  13.657,  13.364,  12.988,  11.754,  11.732,  11.898,  12.179/
      data (Longitude(i),i= 6481, 6560)/
     .  12.490,  12.737,  12.989,  13.235,  13.392,  13.685,  13.952,  14.267,
     .  14.509,  14.528,  14.528,  14.448,  14.449,  14.500,  14.469,  14.642,
     .  14.858,  14.874,  14.982,  15.156,  15.241,  15.445,  15.705,  16.106,
     .  16.499,  16.488,  16.739,  16.980,  17.148,  17.336,  17.593,  17.885,
     .  18.221,  18.340,  18.188,  18.079,  18.400,  18.313,  18.492,  18.491,
     .  19.101,  19.529,  20.137,  20.662,  21.269,  21.871,  22.368,  22.974,
     .  23.567,  24.170,  24.756,  25.314,  25.705,  25.706,  26.204,  26.796,
     .  27.316,  27.804,  28.251,  28.695,  29.098,  29.471,  29.934,  30.298,
     .  30.600,  30.864,  31.063,  31.063,  31.300,  31.680,  32.146,  32.454,
     .  32.602,  32.746,  32.884,  32.893,  32.892,  32.927,  32.546,  32.858/
      data (Longitude(i),i= 6561, 6640)/
     .  33.304,  33.822,  34.343,  34.484,  34.485,  34.997,  35.369,  35.534,
     .  35.480,  35.504,  35.540,  35.296,  35.118,  35.121,  35.121,  34.777,
     .  34.754,  34.590,  35.107,  35.489,  35.852,  36.253,  40.738,  40.647,
     .  40.599,  40.591,  40.477,  40.549,  40.487,  40.457,  40.547,  40.438,
     .  36.253,  36.590,  36.949,  37.280,  37.747,  38.224,  38.723,  39.193,
     .  39.645,  40.007,  40.299,  40.622,  40.642,  40.738,  40.438,  40.056,
     .  39.740,  39.632,  39.445,  39.286,  39.446,  39.350,  39.336,  38.989,
     .  38.787,  38.974,  39.096,  39.205,  39.204,  39.570,  39.809,  39.961,
     .  40.166,  40.486,  40.885,  41.315,  41.559,  41.557,  41.836,  42.105,
     .  42.449,  42.796,  42.894,  42.891,  43.234,  43.579,  43.967,  44.336/
      data (Longitude(i),i= 6641, 6720)/
     .  44.743,  45.173,  45.611,  46.040,  46.392,  46.756,  47.096,  47.453,
     .  47.491,  47.491,  47.819,  48.111,  48.385,  48.681,  48.944,  49.095,
     .  49.311,  49.588,  49.587,  49.819,  50.102,  50.326,  50.610,  50.817,
     .  50.904,  51.182,  51.413,  51.413,  51.122,  51.096,  51.150,  50.640,
     .  50.278,  49.785,  49.287,  48.925,  48.924,  48.416,  47.930,  47.411,
     .  46.968,  46.506,  45.999,  45.490,  45.033,  44.524,  44.055,  43.678,
     .  43.401,  43.101,  43.120,  43.379,  43.105,  42.681,  43.101,  38.612,
     .  38.831,  39.020,  39.161,  43.121,  42.729,  42.347,  42.023,  41.612,
     .  41.310,  41.252,  41.252,  40.763,  40.290,  40.057,  39.566,  39.288,
     .  39.188,  39.159,  38.611,  38.137,  37.792,  37.412,  37.280,  37.198/
      data (Longitude(i),i= 6721, 6800)/
     .  37.230,  37.122,  36.977,  36.841,  36.565,  33.473,  33.472,  33.256,
     .  33.256,  11.279,  11.279,  11.057,  11.057,   8.947,   8.947,  -3.027,
     .  -3.027, -13.455, -13.455, -14.088, -14.423, -13.902, -14.088, -15.818,
     . -15.818, -16.236, -16.758, -16.202, -16.236, -17.269, -17.269, -18.057,
     . -18.057, -17.783, -17.783, -16.020, -16.020, -16.997, -16.997, -16.371,
     . -16.371, -24.701, -24.701, -24.468, -24.468, -23.728, -23.728, -23.216,
     . -23.216, -22.909, -22.909, -22.991, -22.991, -24.425, -24.425, -25.046,
     . -25.046, -25.317, -25.317, -16.336, -16.336, -15.985, -15.985, -15.471,
     . -15.471, -15.942, -15.942, -16.239, -16.239, -16.151, -16.151, -16.059,
     . -16.059, -15.981, -15.981, -15.946, -15.946, -15.661, -15.661, -12.885/
      data (Longitude(i),i= 6801, 6880)/
     . -12.885,   5.634,   5.634,   6.683,   6.683,   7.424,   7.424,   8.919,
     .   8.744,   8.902,   8.919,  11.754,  11.754,  32.972,  32.972,  35.476,
     .  35.476,  43.290,  43.478,  43.274,  43.290,  43.652,  43.652,  44.207,
     .  44.207,  45.097,  45.097,  45.283,  45.283,  46.280,  46.280,  46.448,
     .  46.448,  39.722,  39.722,  39.189,  39.428,  39.356,  39.189,  39.696,
     .  39.678,  39.767,  39.696,  39.669,  39.669,  41.003,  41.003,  39.978,
     .  39.978,  55.710,  55.710,  55.712,  55.713   55.464,  55.464,  56.284,
     .  56.284,  56.532,  56.532,  56.564,  56.564,  57.750,  57.333,  57.750,
     .  63.478,  63.478,  48.217,  48.217,  42.741,  42.741,  45.614,  45.051,
     .  44.542,  44.073,  43.852,  43.663,  43.746,  43.597,  43.319,  43.287/
      data (Longitude(i),i= 6881, 6960)/
     .  43.421,  43.758,  43.946,  44.250,  44.251,  44.251,  44.481,  44.465,
     .  44.231,  44.089,  44.004,  43.937,  44.209,  44.416,  44.753,  45.209,
     .  45.699,  46.223,  46.649,  47.171,  47.261,  47.286,  48.433,  48.301,
     .  48.116,  47.915,  47.816,  47.674,  47.537,  47.322,  47.127,  46.689,
     .  46.131,  45.653,  45.614,  47.287,  47.754,  47.956,  47.890,  48.403,
     .  48.816,  48.940,  49.039,  49.274,  49.274,  49.544,  49.800,  49.976,
     .  50.140,  50.206,  50.239,  50.486,  50.368,  50.166,  50.166,  49.903,
     .  49.688,  49.861,  49.590,  49.447,  49.460,  49.365,  49.171,  48.999,
     .  48.834,  48.654,  48.502,  48.433,  51.759,  51.759,  50.223,  50.223,
     .  37.654,  37.654,  -9.892,  -9.892, -12.210, -12.210,  -5.715,  -5.715/
      data (Longitude(i),i= 6961, 7040)/
     . -14.392, -14.392,  30.532,  30.502,  30.869,  31.267,  31.241,  30.840,
     .  30.531,  29.046,  29.122,  29.287,  29.369,  29.570,  29.596,  29.806,
     .  29.920,  29.945,  29.945,  29.861,  30.317,  30.561,  30.616,  30.884,
     .  31.146,  30.644,  30.463,  30.299,  30.299,  30.185,  29.805,  29.492,
     .  29.320,  29.340,  29.149,  29.100,  29.237,  29.046,  27.022,  27.326,
     .  27.569,  27.968,  28.430,  28.954,  28.482,  28.051,  27.614,  27.249,
     .  27.022,  33.926,  34.071,  34.201,  34.215,  34.325,  34.054,  34.179,
     .  34.334,  34.424,  34.544,  35.048,  34.857,  34.809,  34.821,  34.708,
     .  34.959,  34.963,  34.963,  34.663,  34.657,  34.553,  34.308,  33.928,
     .  31.613,  31.752,  31.848,  31.792,  31.976,  32.254,  32.254,  32.735/
      data (Longitude(i),i= 7041, 7120)/
     .  33.213,  33.714,  34.022,  34.186,  34.186,  34.678,  34.425,  34.131,
     .  33.816,  33.566,  33.203,  33.704,  33.432,  33.426,  33.426,  32.952,
     .  32.832,  32.665,  32.161,  31.844,  31.631,  31.614,  37.325,  37.001,
     .  37.438,  37.451,  37.325,  49.827,  49.827,  35.922,  35.911,  36.068,
     .  35.590,  34.987,  34.370,  33.975,  33.368,  32.806,  32.805,  32.277,
     .  31.790,  31.228,  30.601,  30.523,  29.915,  29.295,  29.014,  28.622,
     .  28.622,  28.019,  27.406,  28.012,  27.449,  27.222,  27.244,  26.674,
     .  26.349,  26.894,  27.015,  26.628,  26.074,  26.073,  26.168,  26.638,
     .  27.269,  27.927,  28.564,  29.123,  29.778,  29.158,  29.734,  30.281,
     .  30.281,  30.921,  31.487,  32.073,  32.656,  33.293,  33.968,  34.641/
      data (Longitude(i),i= 7121, 7200)/
     .  35.021,  35.021,  35.418,  36.087,  36.462,  37.117,  37.780,  38.390,
     .  39.036,  39.701,  39.997,  35.896,  35.821,  35.922,  35.896,  35.973,
     .  35.973,  35.649,  35.448,  35.213,  35.100,  35.100,  34.914,  34.782,
     .  34.547,  34.490,  34.490,  34.219,  34.219,  33.684,  33.102,  32.769,
     .  32.770,  32.306,  32.580,  32.721,  33.071,  33.220,  33.564,  33.564,
     .  33.931,  34.448,  34.516,  34.516,  34.677,  34.891,  34.961,  34.961,
     .  34.979,  34.978,  34.961,  34.961,  34.841,  34.734,  34.708,  34.708,
     .  35.233,  35.233,  35.518,  35.822,  36.163,  36.436,  36.695,  37.075,
     .  37.279,  37.424,  37.843,  37.843,  38.298,  38.636,  38.789,  39.063.
     .  39.037,  39.088,  39.191,  39.504,  39.569,  39.568,  40.006,  40.456/
      data (Longitude(i),i= 7201, 7280)/
     .  40.797,  41.049,  41.213,  41.509,  41.784,  42.203,  42.357,  42.716,
     .  42.821,  42.696,  42.795,  42.955,  43.076,  43.255,  43.228,  43.229,
     .  43.476,  43.931,  44.436,  44.934,  45.393,  45.799,  46.313,  46.823,
     .  47.314,  47.742,  48.253,  48.759,  49.073,  49.183,  49.182,  49.639,
     .  50.149,  50.622,  51.130,  51.623,  52.076,  52.195,  52.543,  53.031,
     .  53.542,  54.010,  54.531,  55.059,  55.204,  55.204,  55.331,  55.795,
     .  56.320,  56.613,  56.884,  57.393,  57.747,  57.741,  57.813,  57.813,
     .  58.037,  58.476,  50.826,  50.546,  50.239,  49.993,  50.007,  49.686,
     .  49.304,  48.834,  48.632,  48.420,  48.417,  58.476,  58.707,  59.034,
     .  59.429,  59.681,  59.811,  59.336,  59.007,  58.674,  58.143,  57.601/
      data (Longitude(i),i= 7281, 7360)/
     .  57.090,  56.734,  56.454,  56.383,  56.270,  56.271,  56.371,  56.080,
     .  56.081,  55.745,  55.331,  54.938,  54.606,  54.108,  53.630,  53.080,
     .  52.523,  51.994,  51.616,  51.219,  51.218,  51.592,  51.489,  51.388,
     .  50.899,  50.758,  50.826,  48.418,  48.164,  47.802,  47.982,  47.944,
     .  47.944,  48.517,  48.537,  48.538,  48.877,  49.405,  49.986,  50.293,
     .  50.669   50.900,  51.107,  51.264,  34.582,  34.087,  33.716,  33.207
     .  32.598,  32.549,  33.122,  33.736,  34.311,  34.582,  26.011,  26.011,
     .  26.420,  26.420,  26.161,  26.161,  26.361,  26.361,  27.065,  27.065,
     .  27.038,  27.038,  27.355,  27.355,  27.863,  27.863,  28.207,  27.916,
     .  28.207,  27.231,  27.231,  41.861,  41.861,  41.754,  42.178,  41.754/
      data (Longitude(i),i= 7361, 7440)/
     .  52.092,  52.092,  54.221,  53.832,  53.353,  53.854,  54.221,  56.052,
     .  56.052,  58.902,  58.728,  58.900,  58.902,  53.862,  53.862,  54.479,
     .  54.479,  54.467,  54.467,  53.335,  53.335,  50.502,  50.502,  48.186,
     .  48.187,  47.149,  47.149,  47.132,  47.132,  46.814,  46.814,  46.750,
     .  46.751,  46.905,  46.905,  46.898,  46.898,  32.299,  32.299,  32.385,
     .  32.387,  47.630,  47.051,  47.632,  47.630,  32.307,  32.319,  32.319,
     .  32.300,  32.299,  32.355,  32.568,  32.580,  77.567,  77.567,  77.534,
     .  77.534,  70.078,  69.344,  68.765,  68.919,  69.587,  70.349,  69.803,
     .  70.078,  69.326,  69.326,  73.234,  73.234,   3.354,   3.354, 172.661,
     . 172.661, 173.469, 173.469, 177.639, 177.639, 178.694, 179.398, 178.661/
      data (Longitude(i),i= 7441, 7520)/
     . 178.694,-178.784,-178.784,-177.925,-177.925,-177.046,-177.046,-176.549,
     .-176.549,-176.276,-176.276,-176.160,-176.160,-176.032,-176.032,-176.042,
     .-176.042,-175.865,-175.865,-175.718,-175.718,-174.159,-174.706,-174.159,
     .-174.011,-173.191,-174.011,-174.011,-172.407,-172.407,-171.250,-171.250,
     .-170.602,-170.602,-170.104,-170.104,-169.693,-169.693,-169.678,-169.678,
     .-167.795,-168.369,-169.042,-168.625,-167.896,-167.795,-166.258,-166.653,
     .-167.423,-166.816,-166.258,-165.659,-165.659,-165.268,-165.268,-164.935,
     .-164.935,-165.514,-165.514, 190.274, 190.274, 189.857, 189.857, 193.891,
     . 194.386, 193.418, 192.619, 193.530, 193.891, 187.086, 187.086, 190.334,
     . 189.621, 188.506, 189.340, 190.332, 190.334, 190.923, 190.923, 179.663/
      data (Longitude(i),i= 7521, 7600)/
     . 179.663,-155.823,-155.336,-154.971,-155.276,-155.686,-155.886,-155.910,
     .-155.823,-156.270,-156.270,-156.568,-156.568,-156.914,-156.914,-157.175,
     .-157.175,-157.967,-157.967,-159.318,-159.318,-160.057,-160.057,-160.534,
     .-160.534,-161.938,-161.938,-164.697,-164.697,-167.995,-167.995,-171.721,
     .-171.721,-173.947,-173.947,-177.322,-177.322,-177.358,-177.358,-178.294,
     .-178.294,-169.543,-169.543,  28.833,  29.489,  29.661, 351.674, 351.674,
     .  18.984,  18.984,  25.414,  25.414,  23.075,  23.928,  21.619,  21.284,
     .  23.075,  21.996,  21.996,  26.614,  26.614,  28.654,  28.654,  24.852,
     .  27.137,  25.845,  24.147,  21.606,  19.481,  17.880,  20.329,  22.214,
     .  24.747,  24.852,  33.095,  33.095,  10.679,  11.765,  12.248,  13.799/
      data (Longitude(i),i= 7601, 7680)/
     .  14.497,  16.591,  14.643,  16.883,  14.849,  14.406,  16.499,  17.625,
     .  18.373,  19.045,  21.203,  19.822,  18.686,  17.008,  15.886,  16.369,
     .  15.359,  12.632,  10.679,  19.247,  19.894,  19.262,  19.247,  56.808,
     .  57.069,  57.184,  57.429,  57.980,  58.529,  59.038,  59.594,  60.144,
     .  60.696,  61.232,  61.610,  61.610,  62.163,  62.718,  63.270,  63.804,
     .  64.354,  64.906,  65.458,  66.011,  66.438,  66.438,  66.739,  67.070,
     .  67.359,  67.532,  67.531,  67.647,  67.647,  67.857,  67.857,  68.011,
     .  68.011,  68.196,  68.196,  68.741,  68.433,  68.806,  69.289,  69.834,
     .  70.371,  70.023,  69.513,  69.211,  69.211,  69.403,  69.803,  70.142,
     .  70.582,  71.117,  71.606,  71.928,  71.929,  72.258,  72.099,  72.554,
      data (Longitude(i),i= 7681, 7760)/
     .  72.556,  73.101,  72.631,  72.846,  72.885,  72.704,  72.716,  73.047,
     .  73.046,  72.851,  73.014,  73.104,  73.234,  73.306,  73.386,  73.617,
     .  73.890,  74.030,  74.304,  74.511,  74.509,  74.631,  74.731,  74.855,
     .  75.074,  75.397,  75.617,  75.844,  76.021,  76.194,  76.417,  76.395,
     .  76.296,  76.395,  76.416,  76.296,  76.431,  76.693,  77.015,  77.446,
     .  77.914,  78.112,  78.112,  78.384,  78.868,  79.373,  78.925,  79.143,
     .  79.236,  79.236,  79.733,  79.845,  79.839,  79.786,  79.988,  80.225,
     .  80.314,  80.057,  80.141,  80.147,  80.147,  80.049,  80.209,  80.592,
     .  81.108,  81.384,  81.908,  82.334,  82.446,  82.906,  83.334,  83.690,
     .  84.124,  84.123,  84.474,  84.732,  85.153,  85.427,  85.954,  86.435/
      data (Longitude(i),i= 7761, 7840)/
     .  86.808,  86.921,  86.981,  87.471,  87.961,  87.933,  88.079,  88.077,
     .  88.204,  88.624,  88.707,  89.037,  89.037,  89.462,  89.545,  89.870,
     .  89.984,  90.044,  89.915,  90.454,  90.516,  90.538,  90.252,  90.252,
     .  90.629,  90.879,  91.411,  91.757,  91.908,  91.988,  92.209,  92.261,
     .  92.261,  92.507,  92.793,  93.159,  93.687,  93.920,  93.676,  94.058,
     .  94.068,  93.910,  93.871,  94.068,  94.039,  94.308,  94.453,  94.569,
     .  94.457,  94.270,  94.211,  94.613,  94.792,  94.792,  95.144,  95.646,
     .  96.031,  96.186,  96.691,  96.904,  96.881,  96.881,  97.259,  97.342,
     .  97.629,  97.728,  97.820,  97.993,  98.053,  98.096,  98.468,  98.471,
     .  98.470,  98.637,  98.704,  98.654,  98.718,  98.576,  98.525,  98.744/
      data (Longitude(i),i= 7841, 7920)/
     .  98.743,  98.569,  98.406,  98.287,  98.233,  98.704,  99.060,  99.348,
     .  99.734,  99.930, 100.128, 102.096, 101.719, 101.354, 100.848, 100.400,
     . 100.151, 100.480, 100.382, 100.279,  99.957,  99.890,  99.386,  99.377,
     .  99.377,  99.174,  99.174,  99.309,  99.491,  99.675,  99.904,  99.984,
     . 100.084, 100.017,  99.997, 100.497, 100.936, 100.847, 101.246, 101.246,
     . 101.757, 102.156, 102.548, 102.909, 102.912, 100.127, 100.354, 100.384,
     . 100.422, 100.624, 100.769, 101.044, 101.310, 101.477, 101.877, 102.316,
     . 102.559, 102.559, 102.958, 103.401, 103.899, 104.121, 103.922, 103.582,
     . 103.454, 103.450, 103.450, 103.421, 103.460, 103.332, 103.080, 102.669,
     . 102.373, 102.096, 102.912, 103.103, 103.605, 103.546, 104.054, 104.446/
      data (Longitude(i),i= 7921, 8000)/
     . 104.446, 104.859, 104.851, 104.816, 104.814, 104.963, 105.401, 105.825,
     . 106.164, 106.164, 105.955, 106.361, 106.273, 106.781, 106.577, 106.577,
     . 107.088, 107.592, 108.043, 108.484, 108.925, 109.233, 109.212, 109.272,
     . 109.325, 109.325, 109.244, 109.232, 109.078, 108.921, 108.691, 108.387,
     . 108.061, 107.611, 107.232, 107.117, 107.117, 106.738, 106.396, 106.239,
     . 105.848, 105.607, 105.805, 105.974, 106.431, 106.500, 106.500, 106.788,
     . 107.240, 107.632, 107.988, 114.033, 114.220, 107.988, 108.522, 109.059,
     . 109.586, 109.950, 109.951, 109.691, 109.820, 110.269, 110.316, 110.394,
     . 110.909, 111.185, 111.185, 111.707, 112.192, 112.730, 113.241, 113.165,
     . 113.542, 113.492, 113.686, 113.796, 113.959, 113.960, 114.034, 114.221/
      data (Longitude(i),i= 8001, 8080)/
     . 114.501, 114.499, 114.933, 115.470, 115.994, 116.507, 116.821, 117.265,
     . 117.701, 117.788, 117.792, 118.327, 118.782, 119.214, 119.574, 119.235,
     . 119.671, 119.573, 120.106, 120.252, 120.666, 120.849, 120.561, 120.558,
     . 121.112, 121.512, 121.618, 121.910, 121.662, 121.221, 120.680, 120.146,
     . 120.146, 120.685, 121.185, 121.719, 121.588, 121.131, 120.676, 120.090,
     . 119.674, 120.121, 120.693, 121.227, 121.785, 121.467, 121.100, 120.887,
     . 120.640, 120.411, 120.144, 119.600, 124.368, 124.628, 125.268, 125.249,
     . 125.642, 125.000, 124.722, 125.279, 125.909, 126.540, 126.697, 128.366,
     . 127.894, 127.375, 127.531, 128.079, 128.642, 129.173, 129.729, 129.802,
     . 129.882, 130.354, 130.696, 126.694, 126.707, 126.867, 126.243, 126.548/
      data (Longitude(i),i= 8081, 8160)/
     . 126.867, 126.871, 126.506, 126.383, 126.548, 127.156, 127.755, 128.329,
     . 128.906, 128.904, 128.903, 129.357, 129.553, 129.438, 129.419, 129.219,
     . 128.889, 128.566, 128.366, 130.694, 131.233, 131.568, 132.211, 132.678,
     . 133.353, 133.994, 134.554, 135.134, 135.462, 135.848, 136.313, 136.785,
     . 137.268, 137.731, 138.121, 138.402, 138.781, 139.163, 139.606, 140.097,
     . 140.171, 140.169, 140.339, 140.548, 140.635, 140.439, 140.670, 140.902,
     . 141.365, 141.143, 141.063, 141.396, 141.396, 140.742, 140.259, 139.508,
     . 138.653, 138.678, 137.912, 137.471, 136.703, 136.768, 136.078, 135.241,
     . 135.755, 136.348, 137.083, 137.734, 138.149, 138.712, 139.429, 140.028,
     . 140.623, 141.239, 141.881, 142.555, 143.512, 144.503, 145.492, 146.472/
      data (Longitude(i),i= 8161, 8240)/
     . 147.403, 148.362, 149.179, 150.175, 151.164, 151.941, 151.278, 152.238,
     . 153.206, 153.741, 153.743, 154.694, 154.087, 154.499, 155.077, 155.888,
     . 156.376, 156.950, 157.936, 158.966, 160.034, 159.925, 160.237, 161.201,
     . 161.897, 162.772, 163.081, 163.707, 164.793, 165.634, 165.648, 164.556,
     . 164.076, 163.803, 163.501, 162.789, 161.914, 161.404, 160.770, 160.079,
      . 159.614, 159.006, 158.304, 157.489, 156.966, 156.476, 155.923, 155.713,
      . 155.595, 155.579, 155.683, 155.858, 155.971, 156.086, 156.241, 156.451,
      . 156.492, 156.701, 157.426, 157.990, 158.395, 158.419, 158.424, 158.953,
      . 159.732, 159.840, 160.121, 160.783, 161.644, 162.022, 161.718, 161.982,
      . 162.509, 163.218, 162.815, 163.117, 162.281, 162.050, 162.495, 163.008/
       data (Longitude(i),i= 8241, 8320)/
      . 163.390, 163.360, 163.360, 164.323, 165.288, 166.099, 166.118, 166.912,
      . 167.747, 168.766, 169.711, 170.189, 170.618, 171.408, 172.151, 172.806,
      . 173.736, 174.725, 175.600, 176.528, 176.987, 176.976, 177.838, 178.848,
      . 179.591, 179.265, 178.356, 178.395, 177.552, 176.597, 175.429, 176.592,
      . 177.728, 178.909, 179.822, 180.006, 180.630, 180.184, 180.816, 181.132,
      . 181.672, 182.859, 183.997, 184.240, 185.303, 186.143, 186.589, 186.588,
      . 187.662, 187.022, 187.667, 188.720, 189.507, 189.357, 188.529, 187.376,
      . 186.109, 185.749, 185.070, 184.745, 183.860, 182.756, 181.736, 180.616,
      . 179.992, 178.879, 177.620, 176.331, 174.889, 173.737, 173.761, 172.347,
      . 170.920, 170.166, 170.931, 170.623, 169.417, 168.230, 167.816, 166.569/
       data (Longitude(i),i= 8321, 8400)/
      . 165.137, 163.705, 162.271, 161.422, 161.292, 161.001, 160.321, 159.689,
      . 159.733, 158.557, 157.046, 155.537, 154.014, 152.542, 151.652, 151.661,
      . 150.725, 149.389, 149.441, 147.863, 146.563, 145.390, 145.732, 144.114,
      . 145.754, 144.380, 142.710, 141.102, 140.052, 139.697, 140.003, 138.426,
      . 136.868, 136.514, 136.521, 134.938, 133.350, 131.947, 131.400, 130.087,
      . 129.236, 129.069, 127.748, 127.234, 126.489, 125.071, 123.494, 121.924,
      . 120.215, 118.591, 117.526, 115.774, 114.039, 113.586, 114.065, 114.092,
      . 114.092, 113.505, 113.301, 111.648, 110.071, 110.102, 108.491, 106.856,
      . 105.614, 106.114, 107.416, 108.808, 109.961, 111.369, 112.897, 113.704,
      . 112.361, 112.743, 111.279, 109.102, 107.143, 107.265, 105.033, 105.859/
       data (Longitude(i),i= 8401, 8480)/
      . 103.540, 101.868, 100.865, 102.036,  99.892,  99.894,  99.811, 100.177,
      . 100.187,  99.103,  99.659,  97.567,  95.677,  93.871,  92.429,  90.500,
      .  88.789,  86.995,  86.903,  86.734,  87.295,  85.844,  86.783,  86.786,
      .  85.802,  87.012,  85.249,  83.477,  81.679,  80.278,  80.812,  81.132,
      .  82.382,  83.534,  83.184,  83.598,  83.045,  83.182,  83.180,  83.048,
      .  82.195,  83.024,  81.669,  80.686,  79.296,  77.739,  76.281,  76.643,
      .  78.056,  76.916,  76.916,  75.529,  75.213,  75.746,  75.483,  75.097,
      .  73.980,  73.142,  73.768,  74.334,  73.763,  73.704,  73.889,  75.278,
      .  76.681,  77.699,  78.098,  77.473,  78.587,  79.048,  79.048,  77.737,
      .  77.226,  77.317,  76.615,  75.246,  74.427,  74.807,  74.310,  73.864/
       data (Longitude(i),i= 8481, 8560)/
      .  72.961,  72.213,  70.974,  69.743,  69.899,  69.899,  70.804,  71.359,
      .  72.320,  73.074,  73.126,  73.111,  72.478,  72.555,  72.609,  72.768,
      .  72.446,  72.229,  72.719,  72.798,  71.559,  71.557,  69.851,  68.906,
      .  68.566,  67.953,  66.826,  67.321,  67.099,  66.789,  68.096,  68.417,
      .  68.812,  68.650,  68.650,  67.541,  66.464,  65.278,  64.190,  62.837,
      .  61.395,  60.285,  60.935,  59.881,  58.701,  57.553,  56.488,  56.493,
      .  55.154,  54.010,  53.844,  52.472,  51.321,  50.153,  49.064,  47.908,
      .  47.752,  46.711,  46.038,  46.041,  45.305,  45.397,  46.707,  46.160,
      .  45.064,  43.691,  44.210,  44.122,  43.839,  44.431,  44.097,  42.908,
      .  41.741,  41.644,  41.644,  40.603,  39.745,  40.124,  40.519,  39.351/
       data (Longitude(i),i= 8561, 8640)/
      .  38.188,  37.473,  36.506,  37.145,  38.064,  37.655,  37.655,  36.508,
      .  35.748,  34.879,  34.630,  34.698,  34.334,  33.358,  32.562,  31.854,
      .  31.854,  32.981,  34.158,  35.243,  36.427,  37.594,  38.832,  39.999,
      .  40.850,  41.376,  41.031,  40.129,  39.609,  39.611,  38.538,  37.580,
      .  36.452,  35.198,  33.770,  32.403,  32.096,  30.786,  27.861,  28.769,
      .  29.734,  29.001,  28.018,  27.256,  26.299,  25.407,  25.404,  24.411,
      .  23.563,  23.808,  24.087,  24.297,  24.297,  24.302,  24.398,  23.622,
      .  23.078,  22.361,  21.579,  21.290,  21.067,  21.064,  21.241,  21.244,
      .  21.102,  20.636,  19.944,  19.625, 122.067, 121.438, 121.153, 117.787,
      . 118.392, 118.994, 118.953, 119.429, 119.767, 122.066, 122.089, 121.618
       data (Longitude(i),i= 8641, 8720)/
      . 121.488, 121.210, 121.854, 122.276, 122.826, 123.412, 124.065, 124.371,
      . 117.787, 117.568, 117.892, 118.534, 119.147, 119.431, 119.983, 120.538,
      . 120.978, 121.153, 119.600, 119.197, 119.533, 119.920, 120.167, 120.731,
     . 121.264, 121.841, 122.469, 122.693, 122.089, 121.461, 120.941, 120.312,
     . 119.849, 119.767,  29.661,  29.636,  30.137,  29.715,  30.138,  30.138,
     .  30.599,  31.034,  31.189,  31.189,  31.906,  31.738,  31.961,  32.090,
     .  32.798,  33.511,  33.010,  33.170,  33.528,  34.007,  34.515,  35.185,
     .  35.885,  36.463,  35.765,  35.113,  34.829,  35.175,  35.283,  35.283,
     .  35.799,  36.504,  37.193,  37.887,  38.596,  39.308,  38.676,  38.035,
     .  37.738,  37.739,  38.224,  38.052,  37.583,  36.869,  37.200,  37.715/
      data (Longitude(i),i= 8721, 8800)/
     .  38.309,  38.945,  39.431,  39.904,  40.218,  40.218,  40.823,  41.356,
     .  41.613,  41.773,  41.403,  40.800,  10.212,  39.998,  67.870,  67.870,
     .  88.119,  88.119,  88.144,  88.146,  88.664,  89.664,  88.872,  88.873,
     .  73.099,  73.099,  73.124,  73.124,  73.163,  73.163,  73.204,  73.204,
     .  73.240,  73.240,  73.247,  73.247,  73.435,  73.435,  72.941,  72.941,
     .  72.959,  72.959,  72.984,  72.984,  73.037,  73.037,  73.096,  73.096,
     .  73.209,  73.209,  73.379,  73.379,  73.514,  73.514,  73.492,  73.492,
     .  73.354,  73.354,  73.170,  73.170,  73.136,  73.136,  73.387,  73.387,
     .  73.317,  73.317,  73.252,  73.252,  73.290,  73.290,  73.365,  73.365,
     .  73.401,  73.401,  73.518,  73.518,  73.553,  73.553,  73.550,  73.550/
      data (Longitude(i),i= 8801, 8880)/
     .  73.544,  73.544,  73.556,  73.556,  73.568,  73.568,  72.926,  72.926,
     .  72.952,  72.952,  73.104,  73.104,  73.116,  73.116,  73.344,  73.344,
     .  73.354,  73.354,  73.315,  73.315,  72.873,  72.873,  72.900,  72.900,
     .  73.023,  73.023,  73.009,  73.009,  73.354,  73.354,  73.361,  73.361,
     .  73.549,  73.549,  73.559,  73.559,  73.564,  73.564,  73.586,  73.586,
     .  72.978,  72.978,  73.584,  73.584,  73.748,  73.748,  72.929,  72.929,
     .  72.832,  72.832,  72.794,  72.794,  72.693,  72.693,  72.703,  72.703,
     .  72.963,  72.963,  73.345,  73.345,  73.438,  73.438,  73.478,  73.478,
     .  73.494,  73.494,  73.508,  73.508,  73.473,  73.473,  73.497,  73.497,
     .  73.519,  73.519,  73.596,  73.596,  73.638,  73.638,  73.703,  73.703/
      data (Longitude(i),i= 8881, 8960)/
     .  73.703,  73.703,  73.377,  73.377,  73.395,  73.395,  72.979,  72.979,
     .  73.448,  73.448,  73.049,  73.049,  73.104,  73.104,  73.113,  73.113,
     .  73.354,  73.354,  73.323,  73.323,  73.398,  73.398,  73.471,  73.471,
     .  73.594,  73.594,  73.630,  73.630,  72.916,  72.916,  72.932,  72.932,
     .  72.964,  72.964,  72.977,  72.977,  72.987,  72.987,  73.393,  73.393,
     .  73.450,  73.450,  73.429,  73.429,  73.390,  73.390,  73.271,  73.271,
     .  73.271,  73.271,  73.242,  73.242,  73.129,  73.129,  73.127,  73.127,
     .  73.197,  73.197,  73.188,  73.188,  73.049,  73.049,  72.691,  72.691,
     .  73.045,  73.045,  73.070,  73.070,  73.106,  73.106,  73.163,  73.163,
     .  73.182,  73.182,  73.214,  73.214,  73.199,  73.199,  72.905,  72.905/
      data (Longitude(i),i= 8961, 9040)/
     .  72.980,  72.980,  72.913,  72.913,  73.414,  73.414,  72.716,  72.716,
     .  72.177,  72.177,  72.113,  72.113,  73.013,  73.013,  72.791,  72.791,
     .  72.750,  72.750,  72.549,  72.549,  72.650,  72.650,  73.672,  73.672,
     .  73.662,  73.662,  73.647,  73.647,  73.082,  73.082,  72.280,  72.280,
     .  72.329,  72.329,  72.299,  72.299,  72.199,  72.199,  72.433,  72.433,
     .  71.351,  71.351,  71.261,  71.261,  71.318,  71.318,  71.339,  71.339,
     .  71.536,  71.536,  71.586,  71.586,  72.222,  72.222,  72.211,  72.211,
     .  71.966,  71.966,  71.867,  71.867,  71.761,  71.761,  71.737,  71.737,
     .  71.773,  71.773,  71.835,  71.835,  79.794,  79.836,  79.940,  80.096,
     .  80.564,  81.040,  81.496,  81.810,  81.878,  81.690,  81.417,  81.415/
      data (Longitude(i),i= 9041, 9120)/
     .  81.414,  81.194,  80.892,  80.567,  80.177,  80.554,  80.059,  80.074,
     .  79.931,  79.823,  79.794,  79.794,  81.756,  81.757,  79.756,  79.756,
     .  79.678,  79.678,  79.882,  79.882,  89.070,  89.069,  92.813,  92.887,
     .  92.779,  92.874,  92.766,  92.753,  90.390,  90.390,  90.388,  90.388,
     .  90.434,  90.434,  90.593,  90.593,  90.611,  90.611,  90.680,  90.861,
     .  90.572,  90.680,  90.621,  90.621,  90.603,  90.603,  90.588,  90.588,
     .  90.472,  90.472,  91.460,  91.460,  91.196,  91.196,  91.099,  91.099,
     .  90.981,  90.981,  90.994,  90.994,  90.955,  90.955,  91.873,  91.873,
     .  91.934,  91.934,  90.666,  90.666,  92.926,  92.931,  93.415,  93.419,
     .  93.694,  93.697,  93.638,  93.641,  93.396,  93.397,  93.721,  93.721/
      data (Longitude(i),i= 9121, 9200)/
     .  94.657,  94.658,  94.824,  94.826,  99.072,  99.072,  97.871,  97.871,
     .  97.889,  97.889,  97.518,  97.519,  97.665,  97.666,  97.808,  97.811,
     .  97.919,  97.921,  97.928,  97.930,  94.278,  94.281,  98.284,  98.285,
     .  97.837,  97.836,  98.311,  98.312,  98.236,  98.233,  98.470,  98.472,
     .  98.623,  98.618,  97.989,  97.985,  98.067,  98.065,  97.844,  97.843,
     .  97.650,  97.649,  97.470,  97.465,  98.053,  98.050,  98.470,  98.466,
     .  98.412,  98.413,  98.239,  98.237,  98.254,  98.252,  98.166,  98.161,
     .  98.244,  98.244,  98.488,  98.489,  98.498,  98.498,  98.430,  98.430,
     .  97.891,  97.891,  97.943,  97.941,  98.230,  98.229,  98.234,  98.234,
     .  98.061,  98.058,  98.286,  98.284,  98.285,  98.284,  93.040,  93.041/
      data (Longitude(i),i= 9201, 9280)/
     .  93.063,  93.021,  92.992,  92.771,  92.742,  92.638,  92.724,  92.811,
     .  92.956,  93.063,  92.735,  92.733,  93.138,  93.138,  93.020,  93.020,
     .  93.873,  93.873,  92.275,  92.280,  92.686,  92.686,  92.559,  92.561,
     .  92.785,  92.787,  93.099,  93.098,  93.608,  93.607,  93.534,  93.534,
     .  93.558,  93.563,  93.390,  93.391,  93.661,  93.664,  93.695,  93.696,
     .  93.863,  93.868, 100.813, 100.814,  98.414,  98.416,  98.325,  98.320,
     . 100.683, 100.684,  98.635,  98.635,  98.537,  98.543,  98.329,  98.333,
     .  99.400,  99.401,  99.208,  99.214, 100.535, 100.542, 100.081, 100.076,
     . 100.036, 100.038,  99.698,  99.699,  99.682,  99.684,  99.661,  99.663,
     . 102.242, 102.243, 102.310, 102.309, 102.561, 102.560,  99.838,  99.836/
      data (Longitude(i),i= 9281, 9360)/
     .  99.852,  99.853, 100.308, 100.308, 104.188, 104.188, 103.980, 103.992,
     . 103.028, 103.031, 104.010, 104.012, 106.656, 106.659, 106.441, 106.440,
     . 106.767, 106.774, 108.942, 108.943, 112.913, 112.913, 114.374, 114.374,
     . 115.431, 115.431, 106.760, 106.760, 106.911, 106.912, 107.343, 107.344,
     . 107.559, 107.559, 107.559, 107.555, 107.716, 107.721, 107.839, 107.839,
     . 107.742, 107.744, 107.619, 107.620, 107.468, 107.471, 107.794, 107.796,
     . 107.870, 107.869, 111.201, 111.201, 111.762, 111.762, 112.741, 112.741,
     . 112.333, 112.333, 112.346, 112.346, 112.271, 112.271, 112.209, 112.209,
     . 111.701, 111.701, 111.600, 111.600, 111.487, 111.487, 116.707, 116.707,
     . 109.083, 109.083, 109.574, 110.107, 110.620, 110.994, 110.671, 110.478/
      data (Longitude(i),i= 9361, 9440)/
     . 110.100, 109.707, 109.188, 108.705, 108.642, 108.858, 109.308, 109.580,
     . 110.451, 110.456, 110.525, 110.532, 118.091, 118.079, 119.701, 119.701,
     . 121.897, 121.901, 122.124, 122.123, 121.969, 121.970, 121.201, 121.734,
     . 121.195, 121.204, 124.726, 124.732, 126.442, 126.446, 126.379, 126.381,
     . 126.834, 126.369, 126.832, 126.840, 127.931, 127.938, 128.722, 128.728,
     . 130.916, 130.923, 121.276, 121.831, 121.825, 121.646, 121.520, 121.400,
     . 121.129, 120.872, 120.373, 120.164, 120.128, 120.297, 120.557, 120.887,
     . 121.286, 121.498, 121.499, 119.525, 119.526, 118.292, 118.294, 131.654,
     . 132.000, 131.606, 131.185, 130.718, 130.294, 129.753, 129.963, 130.559,
     . 130.592, 134.606, 134.379, 133.792, 133.248, 132.963, 132.451, 132.482/
      data (Longitude(i),i= 9441, 9520)/
     . 132.826, 133.432, 133.872, 134.466, 134.604, 141.138, 141.403, 141.618,
     . 141.876, 142.064, 141.818, 141.468, 141.056, 140.958, 141.021, 140.783,
     . 140.615, 140.761, 140.416, 140.044, 140.106, 139.675, 139.100, 138.489,
     . 138.006, 137.397, 136.841, 136.883, 136.318, 136.027, 135.466, 135.108,
     . 135.321, 134.830, 134.221, 133.668, 133.134, 132.527, 132.158, 131.601,
     . 130.996, 131.360, 131.848, 132.304, 132.676, 133.271, 133.880, 134.475,
     . 135.083, 135.654, 135.959, 136.363, 136.722, 136.734, 137.288, 136.878,
     . 137.266, 137.794, 138.361, 138.768, 139.224, 139.495, 139.805, 140.017,
     . 139.977, 139.988, 140.279, 140.913, 140.906, 141.141, 141.954, 142.409,
     . 142.814, 143.347, 143.924, 144.578, 145.176, 145.115, 145.458, 144.881/
      data (Longitude(i),i= 9521, 9600)/
     . 144.193, 143.650, 143.325, 142.651, 142.078, 141.411, 140.726, 140.802,
     . 140.434, 140.066, 139.842, 140.363, 140.468, 141.095, 141.368, 141.666,
     . 141.786, 141.644, 141.959, 145.313, 145.313, 129.468, 129.473, 129.330,
     . 129.331, 129.773, 129.774, 129.537, 129.538, 129.128, 129.123, 135.004,
     . 135.000, 139.414, 139.420, 139.778, 139.783, 139.573, 139.572, 141.176,
     . 141.180, 140.994, 140.995, 139.446, 139.453, 138.513, 138.313, 138.458,
     . 138.511, 133.292, 133.299, 130.592, 130.391, 130.189, 130.228, 130.602,
     . 130.764, 131.291, 131.457, 131.621, 131.656, 129.973, 129.971, 128.655,
     . 128.652, 131.071, 130.879, 131.049, 131.074, 130.520, 130.524, 129.949,
     . 129.949, 129.872, 129.876, 129.714, 129.715, 130.002, 130.006, 129.690/
```

```
      data (Longitude(i),i= 9601, 9680)/
     . 129.217, 129.685, 129.691, 129.190, 129.197, 128.927, 128.934, 128.556,
     . 128.563, 128.279, 127.944, 128.255, 128.287, 126.737, 126.747, 131.234,
     . 131.238, 125.288, 125.292, 124.091, 124.095, 123.786, 123.792, 122.948,
     . 122.955, 131.259, 131.259, 140.321, 140.321, 142.080, 142.080, 142.189,
     . 142.189, 142.202, 142.202, 142.236, 142.236, 142.129, 142.129, 141.358,
     . 141.358, 141.295, 141.295, 145.273, 145.273, 140.882, 140.882, 146.164,
     . 145.882, 146.161, 146.170, 146.174, 146.868, 146.875, 141.832, 141.844,
     . 141.694, 141.821, 142.201, 142.092, 142.160, 142.149, 142.081, 141.857,
     . 142.150, 142.064, 141.984, 141.914, 141.898, 142.444, 143.178, 143.426,
     . 143.534, 143.067, 142.731, 142.548, 142.736, 142.990, 143.462, 144.193/
      data (Longitude(i),i= 9681, 9760)/
     . 144.650, 144.306, 144.123, 143.893, 143.694, 143.552, 143.371, 143.105,
     . 143.326, 143.324, 133.981, 133.982,  60.493,  60.472, 143.324, 143.230,
     . 142.901, 142.863, 142.659, 142.014, 141.832, 136.671, 136.668, 137.216,
     . 138.049, 137.225, 137.222, 150.455, 150.454, 155.481, 155.479, 148.800,
     . 148.236, 147.613, 147.209, 147.495, 147.923, 148.635, 148.800, 149.469,
     . 150.083, 149.474, 152.198, 152.207, 152.421, 152.423, 152.977, 152.981,
     . 153.242, 153.237, 153.981, 153.974, 154.464, 154.467, 154.594, 154.601,
     . 154.355, 154.355, 155.446, 155.448, 155.236, 155.926, 155.213, 155.227,
     . 156.402, 156.397, 166.241, 166.358, 166.249, 167.433, 168.056, 167.444,
     . 167.440, 163.386, 164.228, 163.391, 163.389, 178.791, 180.004, 181.563/
      data (Longitude(i),i= 9761, 9840)/
     . 181.683, 180.109, 179.982, 178.859, 178.789, 169.412, 168.084, 168.907,
     . 169.409, 162.271, 162.273, 161.693, 161.692, 160.716, 160.719, 137.964,
     . 137.963, 135.418, 135.414, 140.588, 140.448, 141.152, 140.198, 141.843,
     . 143.527, 142.470, 141.165, 146.507, 146.782, 148.301, 150.213, 148.949,
     . 146.995, 146.505, 148.406, 148.397, 139.837, 141.691, 143.621, 142.318,
     . 143.950, 144.951, 143.681, 141.801, 140.983, 139.580, 137.497, 137.284,
     . 137.782, 139.232, 140.062, 135.447, 135.448, 126.680, 128.199, 126.784,
     . 126.676, 128.113, 128.101, 127.318, 129.012, 127.327, 127.323, 128.292,
     . 128.292, 126.510, 127.756, 129.094, 127.897, 126.735, 126.341, 126.497,
     . 122.438, 124.001, 125.323, 126.286, 126.441, 124.981, 123.257, 123.362/
      data (Longitude(i),i= 9841, 9920)/
     . 122.437, 124.511, 124.508, 119.703, 119.698, 115.920, 115.917, 112.813,
     . 112.815, 113.134, 113.119, 112.022, 112.014, 107.219, 107.219, 106.524,
     . 106.520, 106.422, 106.422, 105.993, 105.993,  99.475, 101.654, 104.072,
     . 105.239, 103.629, 101.001, 100.288,  99.491,  99.452,  99.459,  99.965,
     .  97.894,  94.971,  93.763,  94.650,  96.956,  99.497,  99.456,  95.595,
     .  93.062,  92.227,  93.328,  96.091,  97.458,  96.315,  95.595,  90.062,
     .  90.068,  79.208,  79.218,  76.617,  76.618,  91.112,  93.686,  91.088,
     .  91.112,  89.167,  89.157,  96.521,  96.538,  97.420,  97.420,  96.644,
     .  96.644,  96.461,  96.464,  82.654,  82.654,  81.636,  81.650,  82.146,
     .  82.150,  82.134,  82.138,  87.014,  87.027,  85.467,  85.471,  86.206/
      data (Longitude(i),i= 9921,10000)/
     .  86.206,  85.647,  85.641,  84.686,  84.692,  83.968,  83.978,  82.830,
     .  82.829,  82.561,  82.565,  79.161,  79.164,  78.664,  78.676,  76.861,
     .  76.868,  76.182,  76.157,  76.088,  76.081,  75.304,  75.308,  74.096,
     .  74.106,  69.868,  70.618,  70.101,  69.874,  58.950,  58.757,  59.951,
     .  58.936,  59.143,  59.125,  48.231,  48.999,  50.194,  49.466,  48.217,
     .  48.245,  53.001,  53.013,  51.397,  52.490,  52.799,  53.151,  54.666,
     .  56.336,  55.482,  55.374,  55.223,  55.224,  55.866,  56.628,  57.329,
     .  55.848,  54.346,  53.641,  52.569,  51.406,  56.985,  55.356,  54.697,
     .  54.265,  55.927,  57.413,  57.292,  58.442,  58.183,  58.185,  60.007,
     .  60.929,  62.460,  64.262,  66.097,  67.820,  68.888,  67.046,  65.994/
      data (Longitude(i),i=10001,10080)/
     .  65.993,  64.086,  61.978,  60.076,  58.565,  57.110,  55.860,  56.974,
     .  58.796,  58.800,  57.635,  57.635,  55.773,  55.803,  52.316,  52.319,
     .  47.616,  49.154,  47.783,  47.637,  44.953,  47.876,  45.116,  44.966,
     .  59.307,  61.596,  60.056,  59.313,  62.551,  65.036,  63.616,  62.557,
     .  60.063,  60.101,  56.102,  56.104,  54.007,  54.005,  54.428,  57.395,
     .  54.429,  55.471,  55.500,  55.564,  55.556,  56.847,  56.847,  57.909,
     .  57.909,  61.765,  61.777,  62.164,  62.194,  42.541,  42.578,  35.798,
     .  35.812,  23.183,  23.192,  22.634,  22.640,  23.238,  23.253,  24.014,
     .  24.022,  22.994,  22.368,  23.009,  60.471,  60.491,  31.519,  32.168,
     .  31.502,  31.519,  50.116,  50.116,  53.493,  53.493,  53.083,  53.083/
      data (Longitude(i),i=10081,10160)/
     .  59.051,  59.051,  59.782,  59.782,  49.340,  48.928,  48.885,  48.942,
     .  49.310,  49.939,  50.374,  50.896,  51.487,  52.111,  52.715,  53.320,
     .  53.943,  53.894,  53.804,  53.841,  53.983,  53.882,  73.750,  73.753,
     . 103.734, 104.141, 104.478, 104.012, 103.703, 103.735, 100.421, 100.681,
     . 100.458, 100.302, 100.435, 109.486, 109.805, 109.641, 109.482, 109.185,
     . 108.894, 108.285, 107.592, 106.893, 106.277, 105.803, 105.035, 104.232,
     . 104.809, 105.541, 106.010, 106.662, 107.096, 107.643, 108.229, 108.560,
     . 108.944, 109.221, 109.507,  74.190,  74.719,  75.449,  76.177,  76.844,
     .  77.571,  78.264,  78.831,  78.424,  77.698,  76.981,  76.267,  75.555,
     .  74.859,  74.290,  74.086,  74.073,  73.664,  73.641,  74.021,  74.102/
      data (Longitude(i),i=10161,10240)/
     .  61.452,  61.844,  61.297,  61.049,  61.386,  60.883,  60.177,  34.641,
     .  35.554,  35.665,  36.087,  36.359,  35.336,  35.048,  34.289,  34.117,
     .  34.993,  35.334,  34.464,  34.640,  38.122,  38.705,  38.321,  37.638,
     .  37.120,  37.899,  37.689,  38.138,  30.925,  31.713,  32.550,  32.943,
     .  32.519,  31.535,  30.855,  30.447,  30.208,  30.946,  29.328,  29.326,
     .  29.329,  29.327,  60.176,  59.976,  59.515,  33.691,  34.411,  34.919,
     .  35.286,  34.763,  34.193,  33.691,  49.340,  49.415,  49.737,  50.389,
     .  49.938,  49.384,  49.097,  48.725,  48.283,  47.901,  47.535,  47.437,
     .  47.388,  46.989,  46.883,  46.883,  47.288,  47.546,  48.240,  48.827,
     .  49.248,  49.950,  50.572,  51.225,  51.877,  52.609,  53.207,  53.119/
      data (Longitude(i),i=10241,10320)/
     .  53.079,  53.078,  52.739,  52.094,  51.384,  50.974,  51.502,  50.802,
     .  50.857,  51.112,  51.263,  51.846,  52.521,  52.577,  52.463,  52.764,
     .  52.884,  53.473,  53.994,  54.288,  54.754,  54.224,  53.614,  52.966,
     .  52.756,  52.762,  53.417,  53.675,  53.148,  53.744,  53.882,  59.515,
     .  58.820,  58.525,  58.257,  58.201,  58.384,  58.968,  59.658,  60.346,
     .  61.008,  60.967,  61.355,  61.452, 149.161, 149.281, 149.486, 149.701,
     . 150.243, 150.732, 150.756, 151.036, 151.397, 151.894, 151.894, 152.128,
     . 152.499, 152.913, 152.999, 153.072, 153.157, 153.057, 153.357, 153.439,
     . 153.439, 153.567, 153.526, 153.357, 153.265, 153.084, 153.056, 152.939,
     . 152.739, 152.544, 152.547, 152.547, 152.125, 151.667, 151.355, 151.096/
      data (Longitude(i),i=10321,10400)/
     . 150.925, 150.779, 150.455, 150.189, 150.086, 150.078, 150.054, 149.900,
     . 149.753, 149.187, 148.553, 147.932, 147.412, 146.978, 146.350, 146.379,
     . 146.379, 145.885, 145.434, 144.799, 144.994, 144.397, 144.021, 143.549,
     . 143.536, 143.533, 142.964, 142.411, 141.783, 141.162, 140.547, 140.161,
     . 139.767, 139.671, 139.671, 139.774, 139.493, 139.052, 139.509, 139.083,
     . 138.468, 138.279, 138.278, 138.455, 138.435, 138.157, 137.883, 137.711,
     . 137.109, 137.454, 137.491, 137.644, 137.875, 137.920, 137.444, 137.224,
     . 136.689, 136.274, 135.889, 135.961, 135.962, 135.454, 135.334, 135.058,
     . 134.796, 134.263, 134.283, 133.844, 133.254, 132.746, 132.482, 132.483,
     . 131.940, 131.432, 130.844, 130.256, 129.668, 129.079, 128.536, 127.991/
      data (Longitude(i),i=10401,10480)/
     . 127.420, 126.829, 126.239, 126.003, 148.008, 148.267, 148.289, 148.366,
     . 147.938, 147.971, 147.993, 147.993, 147.587, 147.253, 146.756, 146.067,
     . 145.719, 145.359, 145.339, 144.983, 144.752, 144.754, 144.645, 145.252,
     . 145.784, 146.414, 147.027, 147.664, 148.006, 126.003, 125.507, 124.969,
     . 124.409, 124.022, 123.734, 123.198, 122.606, 122.004, 121.401, 120.800,
     . 120.198, 119.622, 119.558, 119.556, 118.953, 118.452, 117.940, 117.327,
     . 116.716, 116.149, 115.696, 115.118, 114.958, 114.989, 114.998, 115.585,
     . 115.651, 115.744, 115.745, 115.556, 115.310, 115.062, 114.939, 114.972,
     . 114.797, 114.578, 114.578, 114.370, 114.103, 114.034, 113.769, 113.426,
     . 113.265, 113.546, 113.531, 113.954, 114.251, 113.942, 113.661, 113.613/
      data (Longitude(i),i=10481,10560)/
     . 113.613, 113.391, 113.454, 113.749, 113.813, 113.699, 113.897, 114.124,
     . 114.499, 114.831, 115.332, 115.757, 116.083, 116.085, 116.551, 117.073,
     . 117.606, 118.069, 118.597, 119.032, 119.565, 119.714, 119.714, 120.245,
     . 120.739, 121.200, 121.519, 121.778, 122.151, 122.192, 122.172, 122.566,
     . 122.859, 122.970, 122.969, 123.257, 123.503, 123.810, 123.504, 123.601,
     . 124.113, 124.618, 124.610, 124.676, 125.181, 125.182, 125.184, 124.826,
     . 125.346, 125.298, 125.601, 125.899, 126.142, 126.653, 126.952, 126.955,
     . 127.421, 127.772, 129.135, 128.086, 128.420, 128.939, 129.455, 129.953,
     . 129.511, 129.590, 129.824, 130.269, 130.143, 130.144, 130.609, 130.969,
     . 131.480, 131.993, 132.482, 132.669, 132.188, 132.699, 133.047, 133.048/
      data (Longitude(i),i=10561,10640)/
     . 133.466, 133.964, 134.454, 134.959, 135.469, 135.888, 135.733, 136.224,
     . 136.404, 136.580, 136.581, 136.910, 136.671, 136.483, 135.978, 136.011,
     . 135.795, 135.517, 135.368, 135.368, 135.708, 136.149, 136.524, 137.015,
     . 137.460, 137.883, 138.304, 138.817, 139.176, 139.608, 139.781, 139.781,
     . 140.296, 140.771, 140.957, 141.234, 141.416, 141.449, 141.633, 141.546,
     . 141.600, 141.508, 141.631, 141.632, 141.720, 141.679, 141.946, 142.098,
     . 142.144, 142.609, 142.788, 142.866, 143.245, 143.247, 143.272, 143.497,
     . 143.591, 143.597, 143.773, 144.282, 144.725, 145.154, 145.341, 145.342,
     . 145.235, 145.361, 145.406, 145.728, 145.978, 146.102, 146.009, 146.334,
     . 146.334, 146.424, 146.873, 147.395, 147.697, 148.220, 148.683, 148.727
      data (Longitude(i),i=10641,10720)/
     . 149.157, 149.161,  96.924,  96.924,  96.836,  96.836, 117.592, 118.085,
     . 118.578, 118.230, 118.692, 119.166, 118.842, 118.430, 117.925, 117.464.
     . 117.464, 117.729, 117.350, 116.845, 116.375, 116.094, 115.836, 115.370,
     . 115.151, 115.145, 115.029, 115.020, 114.089, 113.844, 113.488, 113.181,
     . 112.784, 112.295, 111.800, 111.415, 111.263, 111.019, 110.517, 110.017,
     . 109.650, 109.651, 118.656, 118.660, 118.384, 118.395, 117.341, 117.343,
     . 117.093, 117.098, 116.869, 116.873, 115.168, 115.166, 111.312, 111.312,
     . 115.146, 115.029, 115.019, 114.548, 114.088,  95.238,  95.743,  96.142,
     .  96.641,  97.143,  97.635,  97.968,  98.268,  98.592,  99.007,  99.431,
     .  99.826, 100.002, 100.406, 100.826, 100.942, 100.942, 101.430, 101.904/
      data (Longitude(i),i=10721,10800)/
     . 102.190, 102.520, 103.004, 102.659, 103.140, 103.638, 103.491, 103.271,
     . 103.270, 103.398, 103.807, 104.310, 104.452, 104.606, 104.722, 105.209
     . 105.632, 105.997, 105.881, 105.904, 105.891, 105.903, 105.835, 105.545,
     . 105.544, 105.047, 104.613, 104.111, 103.810, 103.372, 102.958, 102.575,
     . 102.259, 101.897, 101.526, 101.251, 100.903, 100.790, 100.576, 100.411,
     . 100.411, 100.122,  99.808,  99.488,  99.124,  98.953,  98.778,  98.565,
     .  98.138,  97.681,  97.597,  97.250,  96.912,  96.449,  96.099,  95.735,
     .  95.405,  95.226,  95.238,  95.116,  95.121,  95.379,  95.380,  95.464,
     .  96.022,  96.477,  96.457,  97.341,  97.343,  97.150,  97.148,  98.582,
     .  98.578,  98.556,  98.555,  98.502,  98.502,  99.201,  98.783,  98.652/
      data (Longitude(i),i=10801,10880)/
     .  99.068,  99.286,  99.196,  99.858,  99.859, 100.206, 100.203, 100.467,
     . 100.467, 102.381, 102.378,  97.918,  97.415,  97.139,  97.629,  97.914,
     . 104.784, 104.789, 105 261, 105.261, 107.482, 107.482, 106.826, 106.826,
     . 105.889, 106.174, 106.286, 106.762, 106.738, 106.262, 105.941, 105.632,
     . 105.104, 105.380, 105.863, 105.894, 107.709, 108.197, 108.077, 107.592,
     . 107.699, 107.709, 103.713, 103.721, 104.208, 104.215, 104.482, 104.485,
     . 104.611, 104.615, 104.552, 104.556, 104.419, 104.420, 104.352, 104.357,
     . 104.286, 104.281, 104.258, 104.258, 104.854, 104.853, 104.666, 104.667,
     . 104.152, 104.153, 103.963, 103.966, 103.938, 103.941, 103.836, 103.836,
     . 103.524, 103.526, 103.507, 103.507, 103.435, 103.436, 103.296, 103.297/
      data (Longitude(i),i=10881,10960)/
     . 103.167, 102.693, 103.171, 103.166, 103.051, 102.549, 103.049, 103.052,
     . 102.479, 102.478, 102.495, 102.009, 102.495, 102.495, 101.774, 101.774,
     . 105.711, 105.706, 106.235, 106.233, 106.238, 106.239, 106.213, 106.213,
     . 107.940, 107.939, 108.066, 108.062, 107.746, 107.746, 108.778, 107.782,
     . 108.974, 108.973, 107.527, 107.529, 109.756, 109.756, 108.951, 108.956,
     . 115.849, 115.845, 116.270, 116.313, 116.167, 116.271, 116.429, 116.429,
     . 118.575, 118.575, 117.548, 117.555, 117.231, 117.235, 117.782, 117.786,
     . 117.609, 117.611, 117.685, 117.688, 109.650, 109.187, 109.023, 108.846,
     . 108.916, 109.185, 109.312, 109.642, 110.023, [illegible].908, 110.100, 110.279,
     . 110.279, 110.609, 111.088, 111.588, 111.836, 112.322, 112.826, 113.326
      data (Longitude(i),i=10961,11040)
     . 113.783, 114.268, 114.546, 114.546, 114.603, 115.089, 115.558, 115.996.
     . 116.268, 116.772, 116.605, 116.276, 116.602, 117.032, 117.445, 117.442,
     . 117.442, 117.529, 117.614, 118.006, 118.505, 118.992, 118.637, 118.224,
     . 117.874, 117.995, 117.648, 117.328, 117.030, 117.521, 117.590, 108.819,
     . 108.826, 109.904, 110.361, 110.849, 111.355, 111.862, 112.364, 112.872,
     . 113.378, 113.846, 114.349, 114.409, 114.152, 113.645, 113.139, 112.811,
     . 112.612, 112.612, 112.127, 111.657, 111.153, 110.688, 110.466, 109.961,
     . 109.463, 108.955, 108.552, 108.094, 107.606, 107.152, 107.003, 107.003,
     . 106.497, 106.013, 105.822, 105.506, 106.007, 106.492, 106.721, 107.224,
     . 107.701, 108.193, 108.689, 109.195, 109.691, 109.906, 112.695, 112.695/
      data (Longitude(i),i=11041,11120)/
     . 115.291, 115.293, 114.479, 114.981, 115.488, 115.305, 114.841, 114.447,
     . 114.478, 115.548, 115.548, 116.340, 116.617, 116.197, 116.055, 116.345,
     . 117.214, 117.686, 118.188, 117.801, 118.308, 118.809, 119.194, 118.701,
     . 118.199, 117.696, 117.200, 116.785, 117.133, 117.215, 119.082, 119.088,
     . 119.035, 119.541, 120.048, 120.487, 120.832, 120.377, 119.955, 119.496,
     . 119.006, 119.037, 119.451, 119.456, 120.272, 120.779, 121.267, 121.773,
     . 122.259, 122.694, 122.345, 121.847, 121.342, 120.842, 120.334, 119.827,
     . 120.115, 120.276, 123.082, 123.091, 123.404, 123.908, 123.536, 123.410,
     . 123.976, 123.982, 124.472, 124.978, 124.554, 124.475, 121.722, 121.726,
     . 122.854, 123.294, 122.954, 122.856, 123.435, 123.440, 123.411, 123.416/
      data (Longitude(i),i=11121,11200)/
     . 125.096, 125.546, 126.031, 126.511, 126.964, 126.511, 126.064, 125.497,
     . 125.060, 124.656, 124.461, 124.463, 123.985, 123.667, 123.517, 124.026,
     . 124.503, 124.851, 125.096, 125.647, 125.647, 125.968, 126.473, 126.050,
     . 125.974, 127.417, 127.416, 127.794, 127.798, 128.617, 128.624, 129.629,
     . 129.632, 131.646, 131.632, 131.324, 131.227, 131.557, 131.649, 131.036,
     . 131.037, 131.753, 131.756, 120.494, 120.529, 120.465, 120.496, 120.074,
     . 119.575, 119.376, 119.514, 119.607, 119.471, 119.016, 118.765, 118.952,
     . 119.161, 119.299, 119.352, 119.645, 119.842, 119.843, 119.759, 119.867,
     . 120.037, 120.489, 120.822, 121.319, 121.791, 122.289, 122.772, 122.847,
     . 122.847, 123.339, 123.833, 124.297, 124.610, 124.992, 125.058, 124.725/
      data (Longitude(i),i=11201,11280)/
     . 124.431, 123.942, 123.441, 123.071, 123.069, 122.568, 122.066, 121.565,
     . 121.077, 120.586, 120.176, 119.999, 120.119, 120.523, 120.738, 120.738,
     . 121.209, 121.578, 122.077, 122.556, 123.031, 123.392, 122.890, 122.559,
     . 122.183, 121.744, 121.341, 121.341, 121.770, 122.014, 122.334, 122.196,
     . 122.586, 122.890, 122.406, 122.095, 121.597, 121.530, 121.596, 121.596,
     . 121.157, 120.931, 121.080, 120.611, 120.200, 120.409, 120.351, 120.366,
     . 120.310, 120.398, 120.074, 125.301, 125.299, 125.452, 125.449, 125.449,
     . 125.446, 125.658, 125.654, 126.861, 126.860, 126.741, 126.740, 126.878,
     . 126.707, 126.731, 126.879, 121.646, 121.647, 121.914, 121.912, 122.049,
     . 122.049, 122.132, 122.133, 122.364, 122.365, 123.554, 123.166, 122.882/
      data (Longitude(i),i=11281,11360)/
     . 123.384, 123.555, 123.144, 123.142, 123.377, 123.378, 123.853, 123.856,
     . 124.421, 124.921, 124.530, 124.425, 125.413, 125.915, 125.438, 125.416,
     . 125.955, 126.076, 125.888, 125.959, 123.119, 123.122, 123.044, 123.046,
     . 121.977, 121.981, 122.709, 122.630, 122.684, 122.712, 123.076, 123.018,
     . 123.094, 122.677, 122.813, 122.845, 123.073, 126.127, 126.629, 127.105,
     . 127.004, 126.503, 126.096, 126.130, 128.182, 128.685, 129.179, 129.666,
     . 130.147, 130.605, 130.872, 130.368, 129.943, 129.446, 128.952, 128.458,
     . 127.956, 128.079, 128.182, 128.030, 128.038, 128.434, 128.441, 128.580,
     . 128.594, 128.794, 128.794, 129.876, 129.883, 131.419, 131.420, 131.631,
     . 131.634, 131.731, 131.733, 1 2.738, 132.739, 133.176, 132.961, 133.157
      data (Longitude(i),i=11361,11440)/
     . 133.178, 134.121, 134.488, 134.198, 134.117, 134.123, 134.511, 134.755,
     . 134.622, 134 318, 134.489, 134.516, 127.461, 127.964, 127.533, 127.461,
     . 127.559, 127.566, 127.527, 127.832, 127.353, 127.531, 127.235, 127.239,
     . 127.243, 127.243, 127.399, 127.404  127.439, 127.439, 127.339, 127.341,
     . 128.276, 128.383, 128.298, 128.277, 128.904, 128.441, 127.945, 127.893,
     . 128.986, 128.408, 127.987, 127.666, 127.663, 127.663, 127.720, 127.598,
     . 127.415, 127.570, 127.875, 128.005, 128.010, 128.010, 127.694, 128.181,
     . 128.514, 128.701, 128.261, 128.680, 128.901, 129.321, 129.326, 130.821,
     . 131.298, 130.804, 130.302, 130.767, 130.825, 130.635, 130.637, 130.841,
     . 130.849, 130.891, 130.899, 129.875, 129.878, 130.351, 129.969, 130.347
      data (Longitude(i),i=11441,11520)/
     . 130.354, 131.631, 131.640, 133.325, 133.329, 138.553, 139.0    138.884,
     . 138.484, 137.979, 137.854, 138.171, 138.570, 138.821, 138.824, 141.008,
     . 140.615, 140.271, 140.048, 139.568, 139.061, 139.045, 138.746, 139.165,
     . 139.187, 139.187, 138.734, 138.433, 138.400, 138.071, 137.633, 137.176,
     . 136.691, 136.217, 135.746, 135.244, 134.815, 134.815, 134.443, 133.951,
     . 133.699, 133.405, 133.007, 132.730, 132.618, 132.195, 132.683, 133.062,
     . 133.559, 133.679, 133.679, 133.947, 133.448, 132.948, 132.451, 132.006,
     . 131.811, 131.324, 131.285, 131.512, 131.512, 131.968, 132.414, 132.905,
     . 133.342, 133.841, 134.135, 134.101, 134.140, 134.359, 134.462, 134.462,
     . 134.863, 135.316, 135.747, 136.019, 136.345, 136.834, 137.201, 137.200/
      data (Longitude(i),i=11521,11600)/
     . 137.546, 138.049, 138.522, 138.976, 139.434, 139.897, 140.401, 140.862,
     . 141.008, 134.243, 134.247, 134.406, 134.413, 134.576, 134.576, 134.892,
     . 134.894, 135.123, 135.127, 135.499, 135.997, 136.497, 136.003, 135.515,
     . 135.505, 135.457, 135.943, 136.338, 135.838, 135.499, 135.461, 141.007,
     . 141.488, 141.947, 142.411, 142.887, 143.384, 143.828, 144.301, 144.747,
     . 145.127, 145.565, 145.809, 146.036, 146.522, 146.963, 147.460, 147.813,
     . 147.537, 147.033, 146.964, 146.961, 147.137, 147.464, 147.832, 148.204,
     . 148.464, 148.797, 149.298, 149.339, 149.842, 150.049, 150.542, 150.391,
     . 149.902, 150.079, 150.079, 149.583, 149.083, 148.578, 148.070, 147.591,
     . 147.313, 146.901, 146.552, 146.284, 146.264, 146.265, 145.877, 145.386/
      data (Longitude(i),i=11601,11680)/
     . 144.899, 144.436, 143.946, 143.541, 143.130, 142.623, 143.097, 143.397,
     . 142.972, 142.477, 141.971, 141.464, 141.008, 143.227, 143.654, 143.208,
     . 143.227, 143.339, 143.339, 150.875, 150.875, 153.204, 153.674, 153.190,
     . 153.204, 154.033, 154.033, 152.555, 152.555, 151.069, 151.069, 150.772,
     . 151.233, 150.754, 150.772, 150.501, 150.918, 150.440, 150.501, 152.560,
     . 152.560, 151.123, 151.123, 150.238, 150.238, 147.131, 147.131, 147.848,
     . 147.848, 152.181, 152.397, 152.071, 152.022, 151.521, 151.216, 150.752,
     . 150.274, 149.767, 149.318, 148.831, 148.358, 148.815, 149.317, 149.816,
     . 150.016, 150.156, 150.659, 151.045, 151.457, 151.607, 152.043, 152.181,
     . 150.821, 151.217, 151.682, 152.110, 152.497, 152.871, 153.060, 152.662/
      data (Longitude(i),i=11681,11760)/
     . 152.375, 151.952, 151.554, 151.131, 150.821, 150.879, 150.879, 149.954,
     . 150.454, 149.954, 149.954, 149.577, 149.577, 151.995, 151.995, 151.948,
     . 151.948, 152.649, 152.649, 153.228, 153.228, 153.662, 153.662, 154.141,
     . 154.141, 154.806, 154.806, 159.520, 159.520, 154.697, 155.137, 155.424,
     . 155.829, 155.713, 155.266, 154.999, 154.714, 154.697, 154.621, 154.621,
     . 149.519, 149.519, 149.165, 149.165, 147.827, 147.827, 147.392, 146.943,
     . 147.421, 147.392, 144.523, 144.523, 144.243, 144.243, 144.516, 144.516,
     . 142.832, 142.832, 145.992, 145.992, 145.079, 145.079, 120.878, 120.914,
     . 122.291, 122.174, 122.238, 122.466, 122.251, 122.033, 121.551, 121.430,
     . 121.589, 121.665, 121.959, 122.377, 122.894, 123.051, 123.539, 123.696/
      data (Longitude(i),i=11761,11840)/
     . 124.021, 124.092, 123.701, 123.310, 122.927, 122.577, 122.654, 122.188,
     . 121.773, 121.356, 120.904, 120.802, 120.628, 120.116, 120.009, 119.895,
     . 119.769, 120.250, 120.304, 120.435, 120.352, 120.477, 120.574, 121.087,
     . 121.555, 122.085, 122.297, 121.867, 121.872, 122.022, 122.026, 121.974,
     . 121.982, 121.523, 121.525, 121.231, 121.232, 121.445, 121.444, 121.967,
     . 121.972, 121.962, 121.974, 121.957, 121.963, 124.234, 124.327, 124.126,
     . 124.236, 123.263, 123.701, 124.053, 123.581, 123.239, 123.268, 123.649,
     . 123.649, 123.003, 123.334, 122.948, 123.012, 122.642, 122.645, 122.278,
     . 122.280, 122.140, 122.029, 122.112, 122.145, 121.881, 121.889, 120.367,
     . 120.882, 121.349, 121.491, 121.386, 120.916, 120.754, 120.454, 120.378/
      data (Longitude(i),i=11841,11920)/
     . 120.135, 120.146, 124.307, 124.823, 125.323, 125.443, 125.553, 125.061,
     . 124.898, 124.506, 124.275, 124.307, 124.345, 124.946, 125.050, 125.198,
     . 125.038, 124.733, 124.709, 124.381, 124.345, 125.671, 125.697, 125.618,
     . 125.671, 126.095, 126.095, 125.448, 125.872, 126.183, 126.114, 126.350,
     . 126.528, 126.585, 126.334, 126.205, 126.082, 125.848, 125.467, 125.603,
     . 125.646, 125.179, 124.701, 124.251, 124.054, 124.071, 123.989, 123.523,
     . 123.124, 122.684, 122.377, 122.152, 122.047, 122.204, 122.656, 123.029,
     . 123.491, 123.859, 124.328, 124.779, 125.191, 125.446, 125.448, 122.089,
     . 122.091, 120.586, 120.590, 121.075, 121.079, 120.181, 120.190, 118.454,
     . 118.454, 123.649, 123.655, 124.744, 124.745, 124.440, 124.323, 123.828/
      data (Longitude(i),i=11921,12000)/
     . 124.149, 124.444, 123.758, 123.759, 124.059, 124.026, 123.864, 123.577,
     . 123.599, 123.851, 124.009, 124.059, 123.537, 123.368, 123.199, 123.251,
     . 122.744, 122.410, 122.804, 122.881, 123.169, 123.537, 122.707, 122.707,
     . 122.026, 122.464, 122.945, 122.799, 122.401, 121.954, 121.988, 122.062,
     . 121.931, 122.028, 119.904, 120.343, 119.871, 119.910, 119.929, 119.929,
     . 119.869, 119.869, 119.862, 119.873, 117.326, 117.326, 117.084, 117.086,
     . 117.059, 117.058, 119.508, 119.506, 119.600, 119.219, 118.748, 118.571,
     . 118.156, 117.820, 117.388, 117.579, 117.981, 118.345, 118.665, 119.032,
     . 119.323, 119.428, 119.509, 149.885, 149.885, 150.509, 150.509, 151.205,
     . 151.205, 153.054, 153.057, 153.145, 153.241, 153.093, 153.054, 153.411/
      data (Longitude(i),i=12001,12080)/
     . 153.411, 153.394, 153.394, 146.632, 146.632, 147.341, 147.341, 145.343,
     . 145.343, 145.326, 145.326, 137.440, 136.822, 137.190, 137.789, 137.440,
     . 136.188, 136.188, 134.468, 134.468, 133.563, 133.563, 143.883, 143.936,
     . 143.974, 143.883, 148.101, 147.854, 148.319, 148.101, 148.339, 148.339,
     . 148.169, 148.169, 147.352, 147.352, 147.135, 147.135, 145.035, 145.035,
     . 144.900, 144.900, 144.740, 144.740, 148.010, 148.010, 113.149, 112.913,
     . 113.215, 113.149, 113.053, 113.053, 113.110, 113.110, 115.298, 115.298,
     . 124.518, 124.518, 124.344, 124.344, 124.908, 124.908, 125.086, 125.086,
     . 130.033, 130.253, 130.596, 130.091, 130.033, 130.942, 130.508, 130.375,
     . 130.852, 131.353, 131.236, 130.942, 132.567, 132.567, 135.969, 135.969/
      data (Longitude(i),i=12081,12160)/
     . 136.183, 136.183, 136.508, 136.508, 136.129, 136.129, 136.153, 136.153,
     . 136.200, 136.200, 136.364, 136.603, 136.846, 136.332, 136.364, 135.704,
     . 135.704, 136.516, 136.516, 136.657, 136.657, 136.841, 136.841, 137.048,
     . 137.048, 139.141, 139.538, 139.158, 139.141, 139.415, 139.415, 142.135,
     . 142.135, 142.271, 142.271, 142.146, 142.146, 142.300, 142.300, 146.221,
     . 146.221, 146.799, 146.799, 148.881, 148.881, 148.967, 148.967, 149.053,
     . 149.053, 132.540, 132.540, 136.728, 136.728, 167.971, 167.971, 159.064,
     . 159.064, 174.012, 174.311, 173.737, 173.220, 173.023, 172.378, 172.105,
     . 171.879, 171.419, 171.196, 171.034, 170.376, 169.847, 169.518, 169.512,
     . 168.889, 168.259, 167.852, 167.285, 167.140, 166.709, 167.051, 167.778/
      data (Longitude(i),i=12161,12240)/
     . 168.346, 169.072, 169.772, 170.253, 170.760, 170.914, 170.576, 171.173,
     . 171.173, 171.342, 171.928, 171.699, 171.080, 171.708, 172.246, 172.940,
     . 172.727, 172.042, 172.731, 173.150, 173.517, 173.912, 174.286, 174.019,
     . 167.484, 167.484, 167.694, 167.690, 168.002, 167.714, 167.798, 168.012,
     . 166.624, 166.623, 166.288, 166.292, 169.210, 169.209, 178.838, 178.833,
     . 179.079, 179.077,-176.224,-176.224,-176.655,-176.655, 166.732, 166.736,
     . 173.785, 173.780, 176.938, 176.834, 176.452, 176.116, 175.683, 175.010,
     . 174.976, 175.211, 174.991, 174.406, 173.840, 174.137, 174.628, 174.724,
     . 174.940, 174.620, 175.230, 175.791, 175.918, 176.120, 176.712, 177.335,
     . 177.817, 178.458, 178.333, 178.180, 177.879, 177.231, 176.950, 176.948/
      data (Longitude(i),i=12241,12320)/
     . 174.940, 174.800, 174.589, 174.339, 174.416, 173.917, 174.193, 173.766,
     . 173.397, 173.197, 172.889, 173.318, 173.935, 174.418, 174.603, 174.794,
     . 174.814, 174.620, 175.526, 175.523, 175.169, 175.169,-178.922,-178.925,
     .-178.610,-178.609,-178.538,-178.546,-177.908,-177.913, 172.145, 172.146,
     . 168.561, 168.559, 153.666, 153.666, 155.449, 155.449, 157.932, 157.932,
     . 157.969, 157.969, 157.156, 157.156, 158.157, 158.157, 158.241, 158.241,
     . 162.913, 162.913, 151.854, 151.854, 151.855, 151.855, 151.577, 151.577,
     . 149.313, 149.313, 149.195, 149.195, 149.186, 149.186, 150.373, 150.373,
     . 150.118, 150.118, 149.665, 149.665, 146.180, 146.180, 143.915, 143.915,
     . 143.060, 143.060, 166.936, 166.936, 169.734, 169.585, 171.743, 171.743/
      data (Longitude(i),i=12321,12400)/
     . 172.765, 172.765, 172.884, 172.884, 172.964, 172.964, 173.003, 173.003,
     . 172.968, 172.968, 172.855, 172.855, 173.118, 173.118, 172.999, 172.999,
     . 173.831, 173.831, 173.636, 173.636, 174.395, 174.395, 174.473, 174.473,
     . 174.738, 174.738, 175.057, 175.057, 175.557, 175.557, 175.975, 175.975,
     . 176.409, 176.409, 176.137, 176.137, 176.312, 176.312, 177.288, 177.288,
     . 177.151, 177.151, 178.703, 178.703, 178.374, 178.374, 179.199, 179.188,
     .-172.202,-172.202,-171.234,-171.234,-171.080,-171.080,-171.710,-171.710,
     . 167.196, 167.196, 167.114, 167.114, 175.971, 175.971, 176.784, 176.784,
     . 168.764, 168.686, 172.026, 171.948, 176.079, 176.079,-176.460,-176.460,
     .-176.633,-176.633, 168.133, 168.133, 153.966, 153.966, 171.925, 171.925/
      data (Longitude(i),i=12401,12480)/
     . 170.840, 170.840, 168.942, 168.942, 168.678, 168.678, 168.558, 168.558,
       167.741, 167.741, 166.833, 166.833, 165.509, 165.509, 166.625, 166.625,
     . 168.220, 168.220, 166.812, 166.812, 165.527, 165.527, 165.272, 165.272,
     . 162.324, 162.324, 167.509, 167.509, 170.136, 170.136, 170.233, 170.233,
     . 169.944, 169.944, 169.856, 169.856, 168.975, 168.975, 171.763, 171.808,
     . 171.577, 171.754, 171.261, 171.389, 171.053, 171.233, 167.491, 167.510,
     . 155.566, 155.566, 155.736, 155.736, 156.046, 156.046, 157.721, 157.721,
     . 156.486, 156.927, 157.278, 156.797, 156.455, 156.486, 156.570, 156.570,
     . 156.549, 156.549, 157.091, 157.091, 157.033, 157.033, 157.097, 157.097,
     . 157.363, 157.363, 157.454, 157.454, 158.216, 158.216, 157.994, 157.994/
      data (Longitude(i),i=12481,12560)/
     . 157.509, 157.833, 157.327, 157.509, 159.605, 159.605, 158.466, 158.922,
     . 159.370, 159.782, 159.278, 158.841, 158.491, 158.466, 158.387, 158.387,
     . 158.276, 158.276, 159.230, 159.230, 159.104, 159.104, 159.811, 159.811,
     . 160.155, 160.155, 160.329, 160.329, 159.705, 160.188, 160.654, 160.184,
     . 159.702, 159.705, 160.720, 160.932, 161.249, 160.775, 160.675, 160.720,
     . 161.364, 161.364, 162.729, 162.729, 161.970, 161.970, 161.314, 161.784,
     . 162.230, 161.722, 161.368, 161.314, 159.803, 159.803, 160.014, 160.465,
     . 159.982, 160.014, 165.794, 165.794, 166.495, 166.495, 166.740, 166.740,
     . 159.251, 159.251, 159.359, 159.359, 159.502, 159.502, 159.724, 159.724,
     . 159.969, 159.969, 163.661, 163.661, 164.221, 164.221, 165.007, 164.578/
      data (Longitude(i),i=12561,12640)/
     . 164.102, 164.310, 164.679, 165.048, 165.480, 165.942, 166.411, 166.910,
     . 166.642, 166.233, 165.787, 165.411, 165.007, 167.516, 167.516, 166.533,
     . 166.533, 166.419, 166.419, 167.388, 167.388, 168.124, 168.124, 171.314,
     . 171.314, 169.894, 169.894, 169.462, 169.462, 169.268, 169.268, 168.526,
     . 168.526, 168.190, 168.190, 168.223, 168.223, 168.168, 168.168, 168.123,
     . 168.123, 167.710, 167.710, 167.443, 167.443, 167.416, 167.416, 167.651,
     . 167.651, 167.314, 167.314, 166.658, 166.658, 166.565, 166.565, 166.772,
     . 166.681, 167.201, 167.058, 166.772, 167.155, 167.155, 167.201, 167.201,
     . 167.422, 167.165, 167.622, 167.422, 177.074, 177.074,-179.957,-180.005,
     .-180.000,-179.957, 178.700, 178.901, 179.404, 179.881, 180.002, 179.998/
      data (Longitude(i),i=12641,12720)/
     . 179.627, 179.152, 178.700, 178.305, 178.305, 179.384, 179.384, 179.423,
     . 179.423, 179.321, 179.321, 178.854, 178.854, 178.000, 177.480, 177.390,
     . 177.814, 178.336, 178.623, 178.231, 178.000, 177.584, 177.584, 177.280,
     . 177.280, 177.146, 177.146, 177.645, 177.645, 178.119, 178.119, 178.494,
     . 178.494, 178.171, 178.171, 179.761, 179.761,-179.864,-179.864, 179.881,
     . 179.881,-179.315,-179.315,-179.143,-179.143,-179.143,-179.143,-178.963,
     .-178.963,-178.315,-178.315,-178.761,-178.761,-178.493,-178.493,-178.966,
     .-178.966,-178.583,-178.583,-178.386,-178.386,-178.202,-178.202,-178.727,
     .-178.727, 179.919, 180.001, 179.998, 179.919, 174.631, 174.631,-178.993,
     .-178.993,-128.331,-128.332,-124.782,-124.783, 134.648, 134.648, 138.192/
      data (Longitude(i),i=12721,12800)/
     . 138.192, 144.886, 144.886, 145.290, 145.290, 145.647, 145.647, 145.809,
     . 145.809, 145.650, 145.650, 145.794, 145.794, 145.850, 145.850, 145.853,
     . 145.853, 145.798, 145.798, 145.688, 145.688, 145.393, 145.393, 144.906,
     . 144.906, 132.218, 132.218, 134.161, 134.161, 134.282, 134.282, 134.450,
     . 134.450,-176.178,-176.178,-172.519,-172.519,-171.741,-171.741,-178.154,
     .-178.154,-175.145,-175.145,-169.927,-169.927,-171.859,-171.859,-170.756,
     .-170.756,-169.518,-169.518,-175.669,-175.669,-174.057,-174.057,-174.057,
     .-174.057,-175.083,-175.083,-174.300,-174.300,-174.934,-174.934,-157.962,
     .-157.962,-159.838,-159.838,-157.964,-157.964,-172.493,-172.493,-171.226,
     .-171.226,-170.561,-170.561,-169.694,-169.694,-174.666,-174.666,-174.071
      data (Longitude(i),i=12801,12880)
     .-174.071,-174.259,-174.259,-165.834,-165.834,-165.422,-165.422,-161.096,
     .-161.096,-161.032,-161.032,-160.993,-160.993,-163.170,-163.170,-159.803,
     .-159.803,-158.944,-158.944,-158.282,-158.282,-158.117,-158.117,-157.719,
     .-157.719,-157.328,-157.328,-151.472,-151.472,-151.458,-151.458,-144.315,
     .-144.315,-130.101,-130.101,-152.874,-152.874,-151.797,-151.797,-143.492,
     .-143.492,-138.635,-138.635,-139.078,-139.078,-138.970,-138.970,-140.073,
     .-140.073,-139.515,-139.515,-140.034,-140.034,-140.654,-140.654,-150.221,
     .-150.221,-150.216,-150.216,-152.400,-152.400,-155.900,-155.900,-154.946,
     .-154.946,-160.011,-160.011,-157.192,-157.192,-159.260,-159.260,-160.380,
     .-160.380,-134.986,-134.986,-140.604,-140.604,-144.626,-144.626,-146.340/
      data (Longitude(i),i=12881,12960)/
     .-146.340,-149.588,-149.588,-149.856,-149.856,-151.011,-151.011,-151.045,
     .-151.045,-151.746,-151.746,-148.232,-148.232,-149.579,-149.579,-149.537,
     .-149.537,-150.657,-150.657,-151.362,-151.362,-149.488,-149.488,-147.670,
     .-147.670,-135.457,-135.522,-135.612,-135.628,-136.199,-136.163,-136.374,
     .-136.411,-136.542,-136.554,-136.310,-136.463,-136.969,-137.068,-138.348,
     .-138.383,-138.424,-138.436,-138.801,-138.837,-138.532,-138.569,-138.928,
     .-138.798,-138.928,-138.848,-140.415,-140.420,-140.685,-140.721,-141.238,
     .-141.268,-140.670,-140.782,-140.821,-140.972,-140.818,-140.653,-140.674,
     .-140.802,-140.107,-140.168,-140.867,-140.905,-142.481,-142.527,-142.542,
     .-142.530,-142.519,-142.514,-142.572,-142.636,-143.064,-143.111,-143.411/
      data (Longitude(i),i=12961,13040)/
     .-143.473,-143.385,-143.549,-143.572,-143.699,-143.815,-143.935,-144.059,
     .-144.120,-144.173,-144.207,-144.254,-144.291,-144.372,-144.435,-144.929,
     .-144.958,-145.001,-144.989,-145.166,-145.237,-145.054,-145.160,-145.433,
     .-145.552,-144.999,-145.000,-145.161,-145.202,-145.227,-145.191,-145.911,
     .-145.841,-145.856,-145.899,-145.922,-145.977,-146.072,-146.059,-146.254,
     .-146.269,-146.220,-146.205,-146.230,-146.239,-145.953,-145.904,-145.710,
     .-145.611,-145.613,-145.482,-145.395,-145.370,-145.189,-145.144,-145.379,
     .-145.432,-145.459,-145.511,-145.575,-145.605,-144.970,-144.957,-146.496,
     .-146.513,-146.609,-146.644,-146.334,-146.379,-147.227,-147.256,-147.347,
     .-147.380,-147.394,-147.449,-147.558,-147.581,-148.041,-148.171,-148.232/
      data (Longitude(i),i=13041,13120)/
     .-148.265,-148.629,-148.689,-148.269,-148.251,-151.722,-151.780,-152.237,
     .-152.268,-152.281,-152.292,-153.920,-153.942,-154.531,-154.530,-145.995,
     .-146.037,-136.645,-136.655,-136.753,-136.758,-138.789,-138.816,-139.212,
     .-139.221,-139.135,-139.160,-141.501,-141.522,-141.912,-141.944,-141.178,
     .-141.221,-141.259,-141.279,-142.259,-142.243,-142.212,-142.225,-142.659,
     .-142.607,-143.104,-143.077,-146.274,-146.314,-146.840,-146.865,-147.794,
     .-147.813,-147.921,-147.918,-148.193,-148.222,-148.695,-148.706,-147.771,
     .-147.743,-144.961,-144.981,-143.555,-143.553, 158.837, 158.837,  51.265,
     .  51.708,  52.248,  52.655,  53.146,  53.602,  54.168,  54.690,  55.201,
     .  55.677,  56.208,  56.766,  56.809,  56.287,  55.798,  56.273,  56.288/

      end
```

```
!!s LatDat
c------------------------------------------------------------------------------
Segment LatDat
      block data LatDat
c------------------------------------------------------------------------------
c     array of latitude values

      include 'LatCom.inc'

      data (latitude(i),i=    1,   80)/
     .  52.838,  53.232,  53.729,  53.534,  53.926,  54.369,  54.829,  55.283,
     .  55.739,  55.923,  69.647,  69.573,  69.254,  68.974,  68.862,  68.792,
     .  69.263,  69.557,  69.655,  69.654,  69.873,  70.084,  69.628,  69.417,
     .  68.991,  69.316,  69.544,  69.842,  69.954,  69.954,  70.229,  69.824,
     .  69.412,  69.849,  69.427,  69.646,  69.823,  48.011,  48.439,  48.930,
     .  49.430,  49.416,  49.701,  49.701,  50.163,  49.743,  50.193,  50.673,
     .  50.459,  50.961,  50.671,  50.952,  50.951,  50.907,  51.131,  51.617,
     .  51.646,  51.700,  52.199,  52.636,  52.310,  52.671,  52.837,  55.922,
     .  55.421,  54.944,  55.337,  55.830,  55.858,  56.361,  56.804,  57.088,
     .  57.566,  58.041,  58.517,  58.518,  58.374,  58.824,  59.313,  58.814
      data (latitude(i),i=   81,  160)/
     .  58.324,  58.572,  58.966,  58.512,  58.403,  58.718,  59.090,  59.380,
     .  59.543,  59.543,  59.846,  59.770,  59.794,  60.022,  60.089,  59.997,
     .  60.291,  60.472,  60.689,  61.012,  61.278,  61.278,  60.857,  60.362,
      .  59.926,  59.842,  59.480,  59.313,  59.716,  60.183,  60.686,  60.988,
      .  60.937,  61.437,  61.248,  61.012,  60.916,  60.916,  60.502,  60.025,
      .  59.660,  59.375,  58.996,  58.551,  58.099,  57.867,  57.554,  57.549,
      .  57.306,  56.879,  56.650,  56.328,  55.893,  55.572,  55.521,  55.374,
      .  55.138,  55.101,  55.113,  55.112,  55.344,  55.769,  55.991,  55.793,
      .  56.292,  56.604,  56.890,  57.298,  57.483,  57.978,  58.475,  58.904,
      .  59.152,  59.151,  58.794,  58.774,  58.757,  58.943,  58.795,  58.641/
       data (latitude(i),i=  161,  240)/
      .  59.066,  59.565,  60.022,  60.522,  60.038,  59.818,  60.048,  60.329,
      .  60.328,  60.807,  60.773,  60.951,  61.448,  61.489,  61.957,  62.418,
      .  62.877,  63.258,  63.052,  63.214,  63.214,  63.446,  63.705,  64.204,
      .  64.700,  64.703,  64.340,  64.586,  64.503,  64.574,  65.011,  65.414,
      .  65.761,  65.761,  66.017,  66.138,  66.518,  66.157,  66.059,  66.264,
      .  66.696,  66.481,  66.888,  67.033,  67.415,  67.815,  67.835,  67.836,
      .  68.116,  68.603,  68.881,  69.066,  69.504,  69.971,  70.261,  70.599,
      .  70.192,  70.192,  70.689,  70.804,  71.094,  71.186,  70.825,  70.898,
      .  70.812,  70.811,  70.408,  70.499,  70.316,  70.163,  70.068,  70.064,
      .  70.014,  69.699,  69.647,  68.281,  67.889,  67.594,  67.526,  67.088/
       data (latitude(i),i=  241,  320)/
      .  67.144,  66.816,  66.815,  66.948,  66.744,  66.280,  66.121,  65.658,
      .  65.936,  65.935,  65.442,  64.948,  65.282,  65.618,  66.074,  66.387,
      .  66.387,  66.370,  66.516,  66.088,  66.263,  65.787,  65.467,  64.993,
      .  64.764,  64.762,  64.490,  64.076,  63.641,  63.213,  63.002,  63.002,
      .  62.993,  63.314,  63.735,  63.754,  63.755,  63.756,  63.345,  63.025,
      .  62.672,  62.249,  62.035,  62.216,  62.411,  62.806,  62.905,  63.138,
      .  63.138,  63.412,  63.878,  64.197,  64.645,  64.904,  64.902,  64.901,
      .  64.399,  64.594,  64.279,  64.369,  64.856,  65.287,  65.320,  64.904,
      .  65.357,  65.432,  65.431,  65.451,  65.893,  66.386,  66.706,  66.593,
      .  66.131,  66.241,  66.563,  66.563,  66.745,  66.647,  66.742,  67.177/
       data (latitude(i),i=  321,  400)/
      .  67.661,  67.940,  55.002,  55.279,  55.642,  56.061,  56.558,  57.062,
      .  57.550,  58.007,  58.337,  58.455,  58.454,  58.897,  59.272,  59.742,
      .  60.243,  60.741,  60.824,  60.823,  61.320,  61.804,  62.305,  62.543,
      .  62.379,  62.264,  62.356,  62.175,  61.835,  61.834,  61.680,  61.191,
      .  61.019,  61.079,  60.581,  60.080,  60.001,  60.061,  60.062,  59.945,
      .  59.478,  58.986,  58.860,  58.386,  57.987,  57.765,  57.765,  58.003,
      .  58.470,  58.043,  58.417,  58.821,  58.359,  58.862,  59.305,  59.792,
      .  59.792,  60.196,  59.861,  59.379,  59.055,  58.879,  58.878,  58.418,
      .  58.165,  57.768,  57.269,  56.951,  56.951,  56.702,  56.298,  55.865,
      .  55.601,  55.107,  55.330,  55.327,  54.824,  55.206,  54.798,  54.740/
       data (latitude(i),i=  401,  480)/
      .  54.284,  54.111,  53.837,  53.723,  53.297,  53.587,  53.714,  53.714,
      .  54.041,  54.181,  53.748,  53.622,  53.249,  52.774,  52.291,  51.837,
      .  51.549,  69.825,  69.738,  69.401,  69.257,  68.978,  68.881,  68.859,
      .  68.482,  68.175,  67.799,  67.799,  67.692,  67.677,  67.829,  67.880,
      .  67.540,  67.215,  66.712,  67.124,  67.581,  68.082,  68.104,  68.105,
      .  68.430,  68.331,  68.129,  68.627,  68.804,  68.729,  68.257,  68.021,
      .  67.802,  67.724,  67.793,  67.793,  67.734,  67.728,  68.199,  68.474,
      .  68.016,  67.526,  67.183,  66.698,  67.162,  67.659,  68.007,  68.003,
      .  67.541,  67.359,  67.847,  68.317,  68.759,  69.256,  69.759,  69.827,
      .  69.827,  69.809,  69.683,  69.241,  68.820,  68.454,  67.955,  67.550/
       data (latitude(i),i=  481,  560)/
      .  67.053,  66.712,  66.711,  66.419,  66.843,  66.929,  66.612,  66.287,
      .  66.516,  66.029,  65.628,  65.333,  65.611,  65.630,  65.630,  65.858,
      .  65.906,  65.679,  65.314,  65.291,  64.812,  64.375,  64.007,  64.077,
      .  64.129,  64.129,  63.627,  63.649,  63.780,  64.010,  63.764,  63.532,
      .  63.162,  62.790,  62.537,  62.153,  49.143,  48.874,  48.476,  48.274,
      .  48.394,  48.240,  47.739,  47.384,  47.000,  46.743,  46.742,  46.977,
      .  47.333,  47.733,  48.121,  48.439,  48.703,  48.794,  55.272,  55.220,
      .  55.081,  54.577,  54.076,  53.574,  53.072,  52.764,  52.762,  52.339,
      .  51.978,  51.561,  51.114,  51.298,  50.962,  50.724,  50.724,  51.154,
      .  51.615,  51.207,  51.708,  52.151,  52.648,  53.134,  53.248,  53.248/
       data (latitude(i),i=  561,  640)/
      .  53.743,  54.215,  54.658,  54.852,  55.001,  51.549,  51.440,  51.269,
     .  51.019,   50.611,   50.249,   50.206,   50.138,   50.188,   50.189,   50.280,
     .  50.294,   50.306,   50.277,   50.291,   50.215,   49.975,   49.505,   49.286,
     .  49.144,   48.794,   48.993,   49.188,   49.261,   49.164,   48.833,   48.381,
     .  48.069,   48.158,   48.045,   47.998,   47.997,   47.956,   47.681,   47.758,
     .  47.285,   46.786,   46.328,   46.150,   46.108,   46.108,   45.729,   45.679,
     .  45.885,   45.574,   45.115,   45.046,   45.044,   44.859,   44.676,   44.500,
     .  44.212,   43.828,   43.497,   43.549,   43.549,   43.959,   44.448,   44.759,
     .  45.100,   45.111,   45.318,   45.387,   45.854,   45.857,   62.146,   61.910,
     .  61.480,   60.990,   60.527,   60.036,   59.533,   59.031,   58.607,   58.740/
      data (latitude(i),i=  641,  720)/
     .  58.743,   58.741,   58.248,   57.765,   57.283,   57.154,   57.228,   56.975,
     .  56.847,   56.516,   56.131,   55.903,   55.695,   55.398,   55.259,   55.272,
     .  45.856,   45.523,   45.268,   45.103,   45.151,   44.656,   44.477,   44.347,
     .  44.346,   44.356,   43.965,   43.734,   43.433,   43.011,   42.519,   42.056,
     .  41.778,   41.828,   41.828,   41.559,   41.478,   41.322,   41.277,   41.164,
     .  40.949,   40.613,   40.510,   40.508,   40.039,   39.583,   39.168,   39.298,
     .  39.791,   39.289,   38.820,   38.452,   38.452,   38.020,   37.551,   37.132,
     .  37.631,   38.131,   38.461,   38.959,   39.370,   39.302,   38.801,   38.301,
     .  38.416,   38.334,   38.333,   38.145,   37.723,   37.989,   37.616,   37.129,
     .  37.254,   36.857,   36.698,   36.546,   36.547,   36.064,   36.537,   36.102/
      data (latitude(i),i=  721,  800)/
     .  35.696,   35.696,   35.399,   34.908,   34.607,   34.753,   34.752,   34.313,
     .  33.894,   33.822,   33.465,   33.151,   33.151,   32.792,   32.524,   32.536,
     .  32.037,   31.581,   31.363,   31.362,   30.861,   30.364,   29.874,   29.401,
     .  28.948,   28.515,   28.013,   27.864,   27.864,   28.343,   27.854,   27.383,
     .  26.921,   26.419,   25.919,   25.461,   25.185,   25.186,   25.139,   25.578,
     .  25.939,   26.399,   26.711,   26.711,   26.999,   27.403,   27.881,   28.303,
     .  28.807,   29.193,   29.301,   29.301,   29.706,   30.041,   29.908,   29.704,
     .  29.969,   30.262,   30.512,   30.511,   30.385,   30.348,   30.707,   30.350,
     .  30.383,   30.181,   30.361,   30.303,   30.302,   30.138,   29.681,   29.290,
     .  28.927,   28.927,   29.315,   29.305,   29.144,   29.334,   29.335,   29.749/
      data (latitude(i),i=  801,  880)/
     .  29.576,   29.632,   29.778,   29.747,   29.745,   29.541,   29.699,   29.204,
     .  28.848,   28.603,   28.639,   28.476,   28.475,   28.106,   27.613,   27.288,
     .  26.814,   26.335,   26.074,   26.074,   25.968,   32.536,   33.003,   33.003,
     .  33.424,   33.740,   34.056,   34.127,   34.423,   34.475,   34.454,   34.454,
     .  34.925,   35.403,   35.747,   36.163,   36.603,   36.731,   36.731,   37.060,
     .  37.518,   38.007,   38.020,   38.142,   38.022,   38.307,   38.307,   38.706,
     .  39.156,   39.659,   40.101,   40.560,   41.029,   41.531,   41.790,   41.791,
     .  42.279,   42.776,   43.271,   43.740,   44.240,   44.743,   44.920,   44.920,
     .  45.422,   45.925,   46.159,   46.326,   46.694,   46.693,   47.114,   47.604,
     .  48.058,   48.399,   48.399,   48.190,   48.132,   47.907,   47.506,   47.885/
      data (latitude(i),i=  881,  960)/
     .  47.384,   47.868,   48.011,   52.647,   52.647,   53.280,   52.820,   53.207,
     .  53.280,   53.701,   53.701,   52.816,   52.816,   53.909,   53.546,   53.837,
     .  53.909,   53.617,   53.276,   53.546,   53.617,   53.798,   53.798,   54.067,
     .  54.067,   55.014,   55.014,   53.229,   52.726,   52.366,   52.654,   53.033,
     .  53.229,   54.124,   53.646,   53.168,   53.464,   53.940,   54.110,   53.640,
     .  54.022,   54.124,   69.578,   69.578,   69.100,   69.484,   68.996,   69.100,
     .  50.697,   50.816,   50.568,   50.121,   50.113,   49.739,   49.377,   48.967,
     .  49.055,   49.241,   49.241,   48.805,   48.547,   48.355,   48.832,   49.242,
     .  49.463,   49.893,   50.338,   50.465,   50.590,   50.699,   48.604,   48.604,
     .  48.711,   48.711,   48.929,   48.929,   49.093,   49.093,   49.792,   49.792/
      data (latitude(i),i=  961, 1040)/
     .  50.226,   50.226,   50.290,   50.290,   50.431,   50.431,   50.821,   50.821,
     .  49.390,   49.390,   51.712,   51.712,   52.048,   52.048,   52.421,   52.005,
     .  52.434,   52.421,   52.289,   52.289,   52.183,   52.183,   52.791,   52.791,
     .  49.879,   49.879,   62.526,   62.526,   61.924,   61.924,   61.588,   61.588,
     .  62.774,   62.774,   67.964,   67.941,   67.975,   67.964,   68.142,   67.640,
     .  67.257,   67.372,   67.876,   68.296,   68.233,   68.142,   63.998,   63.998,
     .  63.453,   63.453,   63.257,   63.419,   63.275,   63.257,   62.980,   62.491,
     .  62.250,   62.744,   62.939,   62.980,   62.411,   61.907,   62.334,   62.411,
     .  56.415,   56.415,   56.525,   56.023,   55.893,   56.321,   56.525,   56.499,
     .  56.499,   60.809,   60.809,   62.740,   62.740,   60.561,   60.561,   60.485/
```

```
      data (latitude(i),i= 1041, 1120)/
     .  60.485,  56.911,  56.911,  54.912,  54.912,  74.027,  74.276,  73.991,
     .  73.604,  73.222,  72.906,  72.570,  72.169,  71.672,  71.434,  71.082,
     .  71.512,  71.854,  72.318,  72.498,  72.493,  72.938,  73.438,  73.937,
     .  74.416,  74.528,  74.286,  74.026,  72.603,  72.940,  72.815,  72.313,
     .  72.456,  72.929,  72.740,  72.264,  71.773,  71.608,  71.612,  72.114,
     .  72.606,  73.101,  73.215,  72.838,  72.345,  71.856,  71.379,  71.000,
     .  70.999,  70.593,  70.351,  70.075,  69.847,  69.442,  69.223,  68.832,
     .  68.863,  68.941,  68.942,  69.149,  69.499,  69.063,  68.879,  68.629,
     .  68.623,  68.621,  68.584,  68.467,  68.851,  69.286,  69.281,  69.588,
     .  69.995,  69.995,  70.211,  70.307,  70.255,  70.273,  70.545,  70.676/
      data (latitude(i),i= 1121, 1200)/
     .  70.595,  70.626,  70.998,  71.240,  71.262,  71.262,  71.460,  71.424,
     .  71.626,  72.121,  72.601,  72.952,  73.214,  72.955,  72.602,  64.916,
     .  64.635,  64.174,  63.983,  63.511,  63.667,  63.981,  63.718,  63.327,
     .  63.524,  63.767,  63.767,  63.586,  64.059,  64.560,  65.059,  65.560,
     .  65.327,  65.164,  64.914,  68.291,  68.291,  68.484,  68.484,  67.940,
     .  67.940,  68.114,  68.114,  68.232,  68.232,  68.092,  68.092,  67.973,
     .  67.973,  67.645,  67.645,  67.462,  67.462,  67.196,  67.196,  68.574,
     .  68.574,  68.768,  68.768,  68.369,  68.369,  68.996,  68.996,  69.118,
     .  69.118,  69.198,  69.198,  69.504,  69.504,  70.660,  70.660,  73.527,
     .  73.042,  73.357,  73.762,  73.527,  73.570,  73.570,  69.882,  69.507/
      data (latitude(i),i= 1201, 1280)/
     .  69.069,  68.573,  68.535,  68.800,  69.299,  69.794,  69.882,  69.546,
     .  69.546,  69.566,  69.566,  68.162,  68.162,  69.781,  69.781,  66.251,
     .  66.251,  65.832,  65.832,  66.010,  66.010,  66.000,  66.000,  63.477,
     .  63.477,  63.543,  63.543,  51.612,  51.161,  50.664,  50.234,  49.781,
     .  50.100,  50.008,  50.007,  49.675,  49.493,  49.499,  49.401,  49.254,
     .  49.252,  48.886,  48.396,  48.600,  48.226,  47.730,  48.027,  48.171,
     .  48.169,  47.699,  47.263,  46.773,  46.980,  47.447,  47.896,  47.419,
     .  47.172,  46.866,  46.866,  47.365,  47.501,  47.593,  47.629,  47.667,
     .  47.634,  47.633,  47.565,  48.066,  48.419,  48.919,  49.172,  49.559,
     .  49.556,  50.022,  50.496,  50.904,  51.360,  51.610,  51.605,  53.179/
      data (latitude(i),i= 1281, 1360)/
     .  52.746,  52.862,  53.179,  52.092,  52.092,  49.841,  49.666,  49.394,
     .  49.171,  49.372,  49.716,  49.841,  50.784,  50.784,  49.634,  49.634,
     .  49.736,  49.736,  48.191,  48.191,  47.641,  47.641,  47.071,  47.071,
     .  46.851,  46.366,  45.986,  46.202,  45.706,  45.622,  45.924,  46.372,
     .  46.816,  46.851,  47.641,  47.298,  47.641,  47.037,  46.537,  46.429,
     .  46.478,  45.979,  46.239,  46.398,  46.704,  47.037,  47.918,  47.918,
     .  43.939,  43.939,  55.246,  54.792,  55.219,  55.246,  56.350,  56.028,
     .  55.636,  55.182,  54.688,  55.122,  55.507,  56.003,  56.350,  55.331,
     .  55.331,  56.129,  56.129,  54.986,  54.986,  55.236,  55.236,  55.413,
     .  55.413,  55.954,  55.469,  55.794,  55.954,  56.336,  56.336,  56.480/
      data (latitude(i),i= 1361, 1440)/
     .  56.480,  56.456,  56.456,  56.786,  56.786,  56.978,  56.476,  56.826,
     .  56.978,  56.899,  56.411,  56.819,  56.899,  57.431,  56.974,  56.472,
     .  56.802,  57.284,  57.431,  57.317,  57.317,  58.249,  58.000,  57.993,
     .  57.749,  57.381,  57.784,  58.246,  58.249,  58.089,  58.089,  58.363,
     .  58.048,  57.581,  58.027,  57.550,  57.109,  57.590,  58.088,  58.363,
     .  60.574,  60.574,  60.393,  60.393,  60.286,  59.865,  60.349,  60.286,
     .  58.593,  58.593,  58.360,  58.009,  58.388,  58.360,  58.002,  57.829,
     .  57.469,  57.081,  56.989,  57.486,  57.321,  57.817,  58.002,  57.136,
     .  57.136,  56.551,  56.551,  57.969,  57.969,  56.604,  56.604,  56.588,
     .  56.588,  55.911,  55.911,  55.278,  55.278,  55.238,  55.238,  55.400/
      data (latitude(i),i= 1441, 1520)/
     .  55.400,  54.965,  54.965,  54.485,  54.485,  54.992,  54.622,  54.471,
     .  54.913,  55.042,  54.992,  58.792,  58.792,  39.540,  39.540,  37.931,
     .  37.931,  35.200,  35.666,  35.195,  35.200,  35.172,  35.172,  35.049,
     .  35.049,  34.915,  34.915,  34.694,  34.694,  34.514,  34.514,  33.893,
     .  33.893,  27.843,  27.372,  27.841,  27.843,  25.478,  25.478,  25.328,
     .  25.328,  24.867,  24.867,  24.813,  24.813,  24.757,  24.757,  24.717,
     .  24.717,  24.745,  24.745,  24.688,  24.688,  24.681,  24.681,  24.591,
     .  24.591,  24.552,  24.552,  26.515,  26.515,  29.743,  29.743,  30.397,
     .  30.336,  30.404,  30.397,  30.241,  30.241,  30.216,  30.216,  29.980,
     .  29.980,  30.116,  30.116,  29.559,  29.559,  29.351,  29.351,  28.476/
      data (latitude(i),i= 1521, 1600)/
     .  28.476,  28.382,  28.382,  28.106,  28.106,  27.820,  27.381,  27.833,
     .  27.820,  27.181,  26.681,  26.195,  26.677,  27.177,  27.181,  24.592,
     .  24.592,  44.801,  44.801,  44.439,  44.439,  41.384,  41.384,  41.465,
     .  41.465,  41.048,  40.835,  40.708,  40.591,  40.933,  40.980,  41.052,
     .  41.048,  45.884,  45.664,  45.833,  45.992,  45.884,  45.990,  45.990,
     .  46.104,  46.104,  46.293,  46.293,  45.734,  45.734,  45.276,  44.823,
     .  45.300,  45.276,  47.336,  47.336,  48.162,  47.856,  48.180,  48.162,
     .  48.839,  48.839,  46.869,  46.869,  46.949,  46.949,  47.062,  47.062,
     .  32.993,  32.993,  33.478,  33.478,  33.279,  33.279,  34.084,  34.084,
     .  34.033,  34.033,  34.057,  34.057,  48.392,  48.392,  65.163,  65.563/
      data (latitude(i),i= 1601, 1680)/
     .  65.987,  66.168,  66.156,  66.441,  66.591,  66.862,  67.028,  66.542,
     .  66.313,  66.578,  66.076,  65.622,  65.792,  65.301,  64.968,  65.432,
     .  65.369,  65.017,  65.163,  53.045,  52.548,  52.125,  51.987,  51.487,
     .  51.034,  50.534,  50.862,  51.359,  51.771,  51.771,  52.218,  52.711,
     .  53.195,  53.649,  54.108,  53.876,  53.396,  53.052,  53.031,  60.882,
     .  60.970,  61.392,  61.537,  61.918,  62.324,  62.487,  62.618,  62.819,
     .  62.658,  62.852,  62.852,  62.942,  62.747,  62.326,  62.043,  62.162,
     .  62.418,  62.654,  62.231,  61.733,  61.403,  61.202,  60.861,  60.888,
     .  43.000,  42.500,  42.043,  41.685,  41.894,  42.374,  42.877,  42.998,
     .  44.502,  44.153,  43.654,  43.344,  43.278,  43.388,  43.331,  43.273/
      data (latitude(i),i= 1681, 1760)/
     .  43.273,  43.264,  43.669,  43.886,  43.976,  43.968,  44.208,  44.365,
     .  44.502,  42.478,  42.663,  42.571,  42.849,  42.846,  42.419,  42.278,
     .  43.021,  43.512,  43.995,  43.810,  44.311,  44.808,  45.304,  45.502,
     .  45.771,  45.773,  45.772,  45.321,  44.835,  44.741,  44.256,  43.757,
     .  43.261,  43.000,  42.998,  43.497,  43.993,  44.479,  44.916,  45.325,
     .  45.758,  45.809,  45.809,  45.974,  45.969,  45.985,  46.483,  46.498,
     .  46.692,  46.685,  46.686,  46.472,  46.658,  46.922,  47.193,  46.888,
     .  46.706,  46.584,  46.584,  46.939,  46.751,  47.224,  47.584,  47.793,
     .  48.011,  48.011,  48.447,  48.703,  48.935,  48.772,  48.746,  48.626,
     .  48.625,  48.155,  47.927,  47.530,  47.039,  46.553,  46.331,  46.284/
      data (latitude(i),i= 1761, 1840)/
     .  46.284,  46.179,  46.123,  45.979,  45.924,  45.489,  45.066,  44.752,
     .  44.752,  44.525,  44.791,  45.193,  44.690,  44.236,  43.741,  43.253,
     .  43.015,  42.238,  42.341,  42.280,  42.122,  42.278,  42.041,  41.857,
     .  41.606,  41.479,  41.450,  41.710,  42.122,  42.238,  41.967,  42.262,
     .  42.478,  41.660,  41.216,  40.756,  41.037,  41.517,  41.660,  66.127,
     .  66.378,  66.765,  67.250,  67.689,  68.047,  68.530,  68.105,  68.098,
     .  68.293,  68.496,  68.496,  68.588,  68.845,  69.291,  69.588,  69.879,
     .  70.172,  70.359,  70.252,  70.068,  70.324,  70.503,  70.439,  70.439,
     .  70.740,  70.952,  71.273,  71.502,  71.541,  72.045,  71.720,  71.539,
     .  71.207,  70.729,  70.433,  70.709,  71.001,  71.001,  71.502,  71.932/
      data (latitude(i),i= 1841, 1920)/
     .  72.254,  72.126,  72.568,  72.619,  72.964,  73.104,  73.432,  73.924,
     .  73.952,  73.952,  73.563,  73.568,  73.452,  73.954,  74.308,  74.457,
     .  74.962,  75.428,  75.554,  75.555,  75.311,  75.724,  75.956,  76.421,
     .  76.824,  76.816,  77.306,  77.414,  77.915,  77.523,  77.525,  78.027,
     .  78.519,  78.964,  79.468,  79.849,  80.095,  80.231,  80.624,  80.721,
     .  81.000,  81.001,  81.418,  81.881,  81.737,  81.535,  81.092,  80.641,
     .  81.129,  81.609,  81.594,  82.025,  81.871,  82.186,  82.253,  82.596,
     .  82.714,  82.712,  83.064,  83.126,  82.970,  83.174,  83.591,  83.553,
     .  83.114,  83.235,  83.200,  83.201,  82.809,  82.362,  82.764,  82.287,
     .  81.805,  82.152,  82.512,  82.008,  81.795,  82.280,  82.262,  82.262/
      data (latitude(i),i= 1921, 2000)/
     .  82.032,  81.599,  81.944,  81.455,  81.136,  80.753,  80.364,  80.062,
     .  80.149,  80.146,  79.685,  79.206,  79.066,  78.735,  78.523,  78.021,
     .  77.849,  77.506,  77.716,  77.716,  77.368,  77.193,  77.165,  77.203,
     .  76.743,  76.273,  76.007,  76.148,  76.373,  76.164,  75.998,  75.998,
     .  75.720,  75.218,  74.849,  74.394,  73.902,  73.420,  73.351,  73.351,
     .  72.864,  72.453,  71.955,  71.454,  71.438,  71.919,  71.613,  71.482,
     .  71.482,  71.166,  71.020,  70.523,  70.649,  70.786,  70.331,  70.017,
     .  70.025,  69.531,  69.051,  68.603,  68.549,  68.191,  68.280,  68.587,
     .  69.060,  69.303,  69.185,  69.586,  69.524,  69.535,  69.536,  69.635,
     .  69.727,  70.198,  69.946,  69.837,  70.236,  70.541,  70.693,  70.477/
      data (latitude(i),i= 2001, 2080)/
     .  70.946,  70.788,  70.628,  70.628,  71.088,  71.588,  71.212,  71.679,
     .  71.306,  71.800,  72.082,  72.582,  72.753,  72.268,  72.128,  72.128,
     .  72.309,  71.848,  72.228,  72.027,  72.520,  73.019,  73.491,  73.729,
     .  73.461,  72.994,  72.983,  72.982,  73.089,  72.771,  72.959,  72.459,
     .  72.097,  72.029,  71.641,  71.142,  71.000,  71.274,  71.689,  72.162,
     .  72.659,  72.999,  72.998,  73.458,  73.816,  73.529,  73.092,  72.611,
     .  72.111,  71.618,  71.262,  71.072,  70.944,  70.795,  70.420,  70.384,
     .  70.010,  69.993,  69.963,  69.922,  69.941,  69.876,  69.876,  70.350,
     .  70.239,  69.742,  69.665,  69.210,  68.842,  68.885,  68.514,  68.215,
     .  67.942,  67.999,  68.461,  68.940,  69.048,  68.554,  68.989,  69.490/
      data (latitude(i),i= 2081, 2160)/
     .  69.501,  69.999,  70.153,  70.153,  70.601,  71.053,  71.526,  71.895,
     .  71.618,  71.149,  71.118,  71.119,  70.622,  70.131,  69.747,  69.483,
     .  68.979,  68.631,  68.249,  68.005,  73.944,  73.802,  73.411,  73.042,
     .  72.732,  72.246,  71.760,  71.424,  71.273,  71.273,  71.696,  72.114,
     .  72.394,  72.832,  72.693,  73.128,  73.450,  73.847,  73.944,  76.168,
     .  75.734,  75.526,  76.028,  76.526,  76.131,  75.761,  75.389,  74.975,
     .  75.096,  75.096,  74.872,  74.605,  74.400,  74.655,  74.971,  75.157,
     .  75.093,  74.975,  75.122,  75.551,  76.017,  76.456,  76.213,  76.167,
     .  76.658,  76.379,  75.923,  75.745,  75.444,  75.403,  75.761,  75.826,
     .  75.622,  75.127,  74.897,  74.896,  74.563,  74.511,  74.509,  74.483/
      data (latitude(i),i= 2161, 2240)/
     .  74.484,  74.552,  74.884,  74.884,  75.187,  75.679,  76.153,  76.314,
     .  76.640,  76.973,  76.606,  76.654,  82.465,  82.955,  82.794,  83.113,
     .  82.934,  82.640,  82.218,  81.849,  81.621,  81.261,  81.101,  81.101,
     .  81.374,  80.877,  80.458,  80.121,  79.828,  79.380,  79.344,  79.366,
     .  79.366,  79.237,  79.189,  78.873,  78.394,  77.967,  77.564,  77.303,
     .  77.607,  77.158,  77.159,  77.231,  76.902,  76.462,  76.171,  76.554,
     .  76.642,  76.371,  76.634,  76.634,  76.446,  76.935,  77.179,  77.632,
     .  77.591,  77.369,  77.817,  77.512,  77.974,  78.457,  78.137,  78.634,
     .  78.866,  78.599,  78.599,  78.599,  78.959,  79.448,  79.841,  80.333,
     .  80.109,  79.685,  80.163,  80.481,  80.583,  80.582,  81.077,  80.699/
      data (latitude(i),i= 2241, 2320)/
     .  80.747,  80.577,  81.021,  80.653,  81.114,  81.038,  81.528,  81.505,
     .  81.506,  81.371,  81.874,  81.934,  82.394,  82.073,  81.834,  82.232,
     .  82.728,  82.881,  82.466,  81.111,  80.904,  80.475,  80.087,  79.596,
     .  79.334,  78.952,  78.510,  78.333,  78.389,  78.842,  79.162,  79.247,
     .  79.247,  79.416,  79.400,  79.874,  80.170,  80.574,  81.049,  81.110,
     .  70.944,  71.044,  70.792,  71.063,  70.566,  70.471,  70.900,  71.063,
     .  72.819,  72.464,  72.407,  72.837,  72.819,  72.912,  72.869,  72.921,
     .  72.912,  73.415,  73.147,  73.045,  73.415,  74.381,  74.381,  75.401,
     .  75.401,  82.073,  82.497,  82.113,  82.073,  81.971,  81.971,  70.316,
     .  70.086,  69.754,  69.354,  69.354,  69.800,  70.289,  70.316,  69.919/
      data (latitude(i),i= 2321, 2400)/
     .  69.919,  73.750,  73.633,  73.515,  73.101,  72.892,  72.748,  73.217,
     .  73.686,  73.750,  69.442,  69.442,  69.390,  68.909,  69.392,  69.390,
     .  69.742,  69.742,  69.790,  69.790,  74.021,  74.170,  74.026,  73.583,
     .  73.130,  72.707,  72.765,  72.264,  71.965,  72.467,  72.946,  73.450,
     .  73.944,  74.021,  74.100,  73.818,  74.100,  75.631,  75.261,  74.758,
     .  74.798,  75.261,  75.631,  75.435,  75.435,  75.934,  75.934,  76.288,
     .  76.288,  76.669,  76.669,  76.722,  76.722,  77.749,  77.317,  77.691,
     .  77.749,  76.114,  75.634,  76.118,  76.114,  77.361,  77.550,  77.050,
     .  76.562,  76.518,  76.036,  75.989,  76.494,  76.914,  77.327,  77.361,
     .  76.885,  76.885,  78.099,  77.605,  77.323,  77.756,  78.073,  78.099
      data (latitude(i),i= 2401, 2480)/
     .  78.059,  78.059,  78.752,  78.295,  78.759,  78.752,  77.893,  77.893,
     .  79.353,  78.897,  78.752,  78.266,  78.268,  78.262,  78.720,  79.020,
     .  79.353,  77.704,  77.461,  77.704,  78.817,  78.492,  78.040,  77.805,
     .  78.280,  78.772,  78.817,  77.654,  77.654,  76.837,  76.837,  76.116,
     .  76.116,  80.151,  79.679,  80.151,  68.706,  68.706,  76.628,  76.591,
     .  76.119,  75.616,  75.116,  74.982,  75.412,  75.521,  75.960,  76.144,
     .  76.566,  76.628,  75.042,  75.042,  76.748,  76.252,  76.748,  76.748,
     .  77.900,  77.900,  77.883,  77.883,  78.938,  78.938,  79.219,  79.219,
     .  80.151,  80.151,  77.454,  77.454,  77.456,  77.456,  71.666,  71.666,
     .  71.301,  71.301,  71.384,  71.384,  69.417,  69.417,  69.372,  69.372/
      data (latitude(i),i= 2481, 2560)/
     .  69.876,  69.876,  73.182,  73.182,  73.119,  73.119,  78.668,  78.668,
     .  78.857,  78.857,  70.481,  70.481,  66.129,  65.873,  66.323,  65.915,
     .  65.572,  65.505,  65.503,  65.003,  64.518,  64.270,  63.989,  63.501,
     .  63.522,  63.522,  63.030,  62.875,  62.401,  61.906,  61.594,  61.594,
     .  61.126,  60.645,  60.584,  60.323,  60.184,  60.098,  60.502,  60.131,
     .  60.594,  60.737,  60.737,  60.514,  61.013,  61.017,  60.919,  61.187,
     .  61.188,  61.417,  61.908,  62.263,  62.735,  63.057,  63.057,  63.201,
     .  63.622,  64.093,  64.453,  64.340,  64.340,  64.784,  65.166,  64.667,
     .  64.283,  64.777,  65.277,  65.728,  65.711,  65.711,  65.695,  65.668,
     .  65.988,  65.869,  66.323,  66.662,  66.832,  66.698,  66.355,  66.836/
      data (latitude(i),i= 2561, 2640)/
     .  67.312,  67.217,  67.187,  67.189,  67.361,  67.314,  67.715,  67.540,
     .  67.767,  67.611,  67.940,  67.993,  68.489,  68.236,  68.193,  65.586,
     .  65.586,  65.532,  65.532,  65.706,  65.706,  64.476,  64.476,  64.202,
     .  64.202,  63.612,  63.612,  63.251,  63.251,  62.790,  62.790,  62.695,
     .  62.695,  62.494,  62.494,  61.769,  61.769,  59.966,  59.966,  59.925,
     .  59.925,  59.915,  59.915,  59.829,  59.829,  59.988,  59.988,  60.114,
     .  60.114,  60.376,  60.376,  60.422,  60.422,  60.434,  60.434,  60.372,
     .  60.372,  60.472,  60.472,  60.772,  60.772,  60.746,  60.746,  60.805,
     .  60.805,  60.824,  60.824,  60.885,  60.885,  60.862,  60.862,  60.915,
     .  60.915,  60.797,  60.797,  60.800,  60.800,  65.574,  65.574,  25.969/
      data (latitude(i),i= 2641, 2720)/
     .  25.479,  25.031,  24.544,  24.044,  23.543,  23.039,  22.542,  22.042
     .  21.646,  22.031,  21.547,  21.099,  20.754,  20.754,  20.346,  19.954,
     .  19.489,  19.063,  18.716,  18.663,  18.364,  18.155,  18.155,  18.295,
     .  18.440,  18.485,  18.682,  18.504,  18.565,  19.029,  19.356,  19.374,
     .  19.374,  19.854,  20.350,  20.843,  21.196,  21.322,  21.398,  21.572,
     .  21.567,  21.446,  21.623,  21.623,  21.201,  20.722,  20.345,  19.853,
     .  19.385,  18.882,  18.398,  18.838,  18.486,  18.486,  18.005,  17.506,
     .  17.008,  16.520,  16.195,  15.894,  15.894,  15.832,  15.723,  15.723,
     .  15.914,  15.814,  15.789,  15.882,  15.944,  15.944,  15.916,  15.876,
     .  15.805,  15.805,  15.466,  15.254,  14.994,  14.994,  14.502,  14.023/
      data (latitude(i),i= 2721, 2800)/
     .  13.550,  13.549,  13.046,  12.544,  12.063,  12.066,  12.065,  11.565,
     .  11.069,  10.925,  10.925,  10.463,  10.065,   9.694,   9.566,   7.227,
     .   7.615,   8.067,   8.224,   8.128,   8.128,   8.478,   8.665,   8.951,
     .   9.141,   9.141,   9.000,   8.933,   8.933,   8.885,   8.885,   8.493,
     .   8.308,   7.999,   7.999,   7.627,   7.277,   7.247,   7.732,   7.896,
     .   7.896,   8.021,   8.199,   8.351,   8.351,   8.072,   8.028,   9.566,
     .   9.186,   8.926,   8.837,   8.837,   8.808,   9.048,   9.201,   9.281,
     .   9.274,   9.375,   9.375,   9.560,   9.412,   9.338,   9.221,   9.221,
     .   8.857,   8.676,   8.028,   8.481,   8.493,   8.965,   8.965,   9.318,
     .   9.515,   9.929,  10.166,  10.279,  10.279,   9.927,   9.852,  10.171/
      data (latitude(i),i= 2801, 2880)/
     .  10.651,  11.075,  11.075,  11.411,  11.733,  12.149,  12.448,  12.431,
     .  12.431,  12.785,  12.986,  12.987,  13.386,  13.406,  13.406,  13.164,
     .  13.241,  13.368,  13.368,  13.505,  13.714,  13.746,  13.746,  13.921,
     .  13.915,  14.077,  14.368,  14.548,  16.187,  16.322,  15.980,  15.825,
     .  15.687,  15.781,  15.934,  16.086,  16.268,  16.526,  16.670,  16.855,
     .  17.030,  17.133,  17.132,  17.367,  17.626,  17.978,  18.019,  18.170,
     .  18.332,  18.733,  18.988,  19.222,  19.573,  19.831,  19.832,  20.282,
     .  20.634,  21.141,  21.604,  22.062,  22.556,  22.930,  23.301,  23.702,
     .  24.076,  24.385,  24.730,  25.158,  25.432,  25.680,  26.179,  26.673,
     .  26.737,  26.736,  27.128,  27.466,  27.911,  28.178,  28.538,  28.934
      data (latitude(i),i= 2881, 2960)/
     .  29.354,  29.814,  30.286,  30.737,  31.233,  31.349,  31.349,  31.513,
     .  31.771,  31.286,  30.808,  30.309,  29.857,  29.530,  29.143,  28.786,
     .  28.366,  27.881,  27.527,  27.118,  27.102,  27.102,  26.618,  26.263,
     .  25.769,  25.353,  24.919,  24.415,  24.289,  23.898,  23.520,  23.056,
     .  23.272,  23.660,  23.985,  24.281,  24.569,  24.999,  25.499,  25.985,
     .  26.094,  26.094,  26.399,  26.701,  26.859,  27.157,  27.491,  27.821,
     .  27.775,  28.158,  28.658,  28.991,  29.380,  29.429,  29.427,  29.734,
     .  30.219,  30.679,  31.127,  31.533,  32.005,  32.456,  32.536,  16.187,
     .  16.031,  15.780,  15.460,  15.154,  15.092,  14.548,  14.906,  15.092,
     .  32.312,  32.312,  21.860,  21.860,  21.747,  21.747,  21.855,  21.855/
      data (latitude(i),i= 2961, 3040)/
     .  21.958,  21.958,  22.413,  22.413,  21.497,  21.497,  21.026,  21.168,
     .  20.916,  21.026,  23.097,  23.097,  22.835,  22.835,  22.731,  22.294,
     .  22.731,  22.666,  22.666,  23.668,  23.192,  23.668,  23.716,  23.716,
     .  24.076,  24.076,  24.644,  24.152,  24.644,  23.456,  23.456,  23.675,
     .  23.675,  24.285,  23.804,  24.285,  25.202,  24.811,  24.324,  24.610,
     .  25.042,  25.202,  24.179,  24.179,  25.063,  25.063,  25.559,  25.264,
     .  24.768,  25.254,  25.559,  26.898,  26.774,  26.390,  25.905,  26.406,
     .  26.839,  26.898,  26.686,  26.704,  26.665,  26.686,  18.090,  18.090,
     .  18.376,  18.482,  18.464,  18.372,  17.975,  17.973,  17.966,  18.376,
     .  18.117,  18.117,  17.771,  17.771,  18.366,  18.366,  18.445,  18.445/
      data (latitude(i),i= 3041, 3120)/
     .  18.742,  18.742,  18.372,  18.372,  18.164,  18.164,  18.068,  18.068,
     .  17.930,  17.930,  17.499,  17.499,  17.402,  17.402,  17.197,  17.197,
     .  17.692,  17.692,  17.167,  17.167,  16.814,  16.814,  16.393,  16.393,
     .  16.048,  16.048,  15.993,  15.993,  15.628,  15.628,  14.859,  14.512,
     .  14.827,  14.859,  19.718,  19.905,  19.911,  19.778,  19.659,  19.336,
     .  19.016,  18.846,  18.367,  18.405,  18.404,  18.392,  18.203,  18.308,
     .  17.852,  18.034,  18.219,  18.156,  18.228,  18.149,  18.284,  18.567,
     .  18.507,  18.445,  18.623,  18.969,  19.469,  19.626,  19.930,  19.776,
     .  19.718,  18.195,  18.195,  17.615,  17.615,  18.091,  18.091,  18.625,
     .  18.625,  18.932,  18.744,  18.922,  18.932,  20.062,  20.062,  17.417/
      data (latitude(i),i= 3121, 3200)/
     .  17.417,  17.414,  17.414,  19.326,  19.326,  19.662,  19.662,  18.263,
     .  18.512,  18.457,  18.350,  18.162,  17.974,  17.833,  18.016,  18.220,
     .  18.263,  21.919,  22.040,  22.512,  22.778,  22.983,  23.030,  23.184,
     .  23.157,  23.026,  23.149,  22.937,  22.679,  22.380,  22.344,  22.123,
     .  21.799,  21.682,  21.334,  21.241,  21.113,  20.689,  20.685,  20.380,
     .  20.200,  20.200,  20.047,  19.888,  19.936,  19.991,  19.937,  19.906,
     .  20.382,  20.686,  20.718,  21.040,  21.524,  21.547,  21.689,  21.852,
     .  22.057,  22.280,  22.196,  22.677,  22.686,  22.548,  22.221,  22.171,
     .  21.863,  21.822,  21.917,  22.800,  22.800,  22.666,  22.666,  22.528,
     .  22.528,  22.404,  22.404,  22.309,  22.309,  22.098,  22.098,  20.602/
      data (latitude(i),i= 3201, 3280)/
     .  20.602,  20.751,  20.751,  20.810,  20.810,  20.894,  20.894,  21.001,
     .  21.001,  21.120,  21.120,  21.601,  21.601,  21.838,  21.641,  21.468,
     .  21.838,  21.339,  21.339,  21.511,  21.511,  25.442,  25.442,  24.369,
     .  24.369,  24.581,  24.581,  25.092,  25.092,  26.065,  26.065,  29.198,
     .  29.198,  28.727,  28.727,  28.684,  28.684,  29.584,  29.217,  29.584,
     .  31.685,  31.685,  28.917,  28.917,  28.068,  28.068,  25.281,  24.778,
     .  25.279,  25.281,  24.530,  24.530,  19.329,  19.329,  18.868,  18.868,
     .  18.351,  18.351,  21.797,  21.797,  21.691,  21.691,  18.789,  18.789,
     .  20.561,  20.561,  18.719,  18.719,  18.530,  18.530,  10.294,  10.293,
     .  17.909,  17.909,  17.300,  17.301,  16.097,  16.097,  16.298,  16.300/
      data (latitude(i),i= 3281, 3360)/
     .  16.418,  16.418,  13.248,  13.248,  13.303,  13.303,  14.368,  14.369,
     .  12.185,  12.185,  11.584,  11.583,   9.435,   9.433,   9.351,   9.349,
     .   9.350,   9.349,   8.417,   8.415,   8.305,   8.304,   8.457,   8.453,
     .   7.579,   7.578,   7.640,   7.639,   7.290,   7.289,   8.131,   8.131,
     .   8.227,   8.227,   8.288,   8.288,  25.426,  24.937,  24.444,  23.943,
     .  24.444,  24.944,  25.437,  25.426,  12.114,  11.855,  11.498,  11.147,
     .  11.411,  11.889,  12.113,   9.262,   9.305,   9.104,   8.954,   8.675,
     .   8.283,   8.663,   8.957,   9.002,   1.947,   2.448,   2.548,   2.859,
     .   3.288,   3.706,   4.108,   4.597,   5.096,   5.578,   6.065,   6.500,
     .   9.002,   9.404,   9.803,  10.295,  10.729,  11.075,  10.986,  11.345/
      data (latitude(i),i= 3361, 3440)/
     . 11.353,  11.356,  11.268,  11.481,  11.738,  12.012,  12.374,  12.355,
     . 11.852,   6.500,   6.869,   7.229,   1.437,   1.091,   0.918,   0.719,
     .  0.218,  -0.187,  -0.665,  -0.990,  -1.487,  -1.715,  -1.714,  -2.203,
     . -2.537,  -2.403,  -2.859,  -3.291,  -3.374,  -3.374,  -3.691,  -4.075,
     . -4.501,  -4.960,  -5.436,  -5.889,  -6.264,  -6.505,  -6.777,  -7.197,
     . -7.644,  -8.043,  -8.090,  -8.091,  -8.502,  -8.970,  -9.418,  -9.885,
     . -10.346, -10.773, -11.264, -11.592, -12.074, -12.444, -12.870, -13.306,
     . -13.806, -13.857, -13.857, -14.342, -14.736, -15.051, -15.435, -15.665,
     . -15.906, -16.199, -16.407, -16.632, -16.825, -17.137, -17.378, -17.810,
     . -18.104, -18.349,   1.947,   1.675,   1.437, -18.348, -18.846, -19.346/
      data (latitude(i),i= 3441, 3520)/
     . -19.835, -20.339, -20.842, -21.341, -21.844, -22.339, -22.838, -23.259,
     . -23.747, -23.843, -40.813, -41.315, -41.771, -41.502, -41.491, -41.857,
     . -42.293, -42.708, -43.212, -43.702, -44.178, -44.496, -44.344, -44.344,
     . -44.805, -45.185, -45.659, -46.160, -46.032, -46.032, -46.447, -46.238,
     . -46.278, -46.278, -46.554, -46.794, -46.838, -47.339, -47.664, -47.838,
     . -47.981, -48.026, -48.528, -48.739, -48.740, -49.187, -49.550, -50.056,
     . -50.141, -50.475, -50.949, -51.447, -51.936, -51.655, -51.656, -51.832,
     . -51.509, -51.726, -52.226, -52.219, -52.672, -53.102, -52.699, -52.535,
     . -52.633, -52.633, -52.737, -53.076, -53.528, -53.043, -52.853, -53.339,
     . -53.825, -53.467, -52.971, -52.630, -52.381, -52.307, -52.381, -23.844/
      data (latitude(i),i= 3521, 3600)/
     . -24.342, -24.842, -25.334, -25.776, -26.276, -26.770, -27.242, -27.740,
     . -28.235, -28.711, -29.195, -29.672, -30.167, -30.588, -31.090, -31.587,
     . -31.612, -31.612, -32.113, -32.613, -33.079, -33.582, -34.024, -34.516,
     . -35.005, -35.444, -35.913, -36.397, -36.852, -37.174, -37.678, -38.159,
     . -38.661, -39.138, -39.637, -40.079, -40.572, -40.812, -52.385, -51.924,
     . -51.592, -51.114, -50.609, -50.234, -49.742, -50.113, -49.708, -49.210,
     . -48.792, -48.449, -48.123, -47.862, -47.862, -47.577, -47.089, -47.047,
     . -46.699, -46.272, -45.794, -45.347, -45.068, -45.012, -44.557, -44.074,
     . -43.572, -43.304, -43.304, -43.059, -42.782, -42.510, -42.864, -42.433,
     . -42.227, -42.184, -41.689, -41.304, -41.304, -40.804, -40.909, -41.152/
      data (latitude(i),i= 3601, 3680)/
     . -41.160, -40.960, -40.474, -39.976, -39.517, -39.056, -38.989, -38.976,
     . -38.917, -38.830, -38.706, -38.548, -38.336, -37.957, -37.544, -37.115,
     . -36.627, -36.275, -35.824, -35.356, -35.000, -34.754, -34.342, -33.841,
     . -33.339, -32.912, -32.448, -32.448, -32.935, -33.348, -33.851, -34.241,
     . -34.458, -34.625, -34.863, -34.789, -34.894, -34.819, -34.560, -34.170,
     . -33.742, -33.741, -33.405, -33.000, -32.521, -32.082, -31.577, -31.259,
     . -30.816, -30.376, -30.749, -31.071, -31.509, -31.871, -31.624, -31.266,
     . -30.858, -30.423, -29.951, -29.500, -29.090, -28.744, -28.362, -27.875,
     . -27.374, -26.876, -26.375, -25.880, -25.400, -25.293, -25.219, -16.007,
     . -15.508, -15.006, -14.506, -14.012, -13.518, -13.048, -12.942, -12.560/
      data (latitude(i),i= 3681, 3760)/
     . -12.138, -11.746, -11.746, -11.272, -10.904, -10.577, -10.236,  -9.862,
     . -9.489,  -9.085,  -8.916,  -8.916,  -8.445,  -7.967,  -7.462,  -6.961,
     . -6.472,  -5.987,  -5.507,  -5.122,  -5.052,  -5.052,  -5.082,  -4.933,
     . -4.665,  -4.387,  -4.041,  -3.701,  -3.407,  -3.171,  -2.922,  -2.808,
     . -2.858,  -2.882,  -2.881,  -2.906,  -2.823,  -2.679,  -2.459,  -2.489,
     . -2.626,  -2.908,  -3.284,  -2.793,  -2.375,  -1.897,  -1.616,  -1.311,
     . -1.299,  -1.113, -25.216, -24.803, -24.550, -24.205, -23.878, -23.763,
     . -23.601, -23.391, -23.353, -23.354, -22.944, -22.898, -23.010, -22.967,
     . -22.947, -22.531, -22.223, -22.031, -21.532, -21.276, -21.276, -20.836,
     . -20.469, -20.002, -19.618, -19.120, -18.617, -18.134, -17.778, -17.278/
      data (latitude(i),i= 3761, 3840)/
     . -16.779, -16.287, -16.007,  -1.113,  -1.032,  -0.804,  -0.647,  -0.719,
     . -1.043,  -1.524,  -1.732,  -2.056,  -2.524,  -2.671,  -2.672,  -2.215,
     . -1.898,  -2.020,  -1.783,  -1.782,  -1.289,  -1.256,  -1.405,  -1.405,
       -1.619,  -1.588,  -1.348,  -1.075,  -1.017,  -1.017,  -0.541,  -0.138,
     .  0.174,   0.532,   0.907,   1.325,   1.768,   1.998,   2.464,   2.941,
     .  3.430,   3.922,   4.325,   4.033,   4.033,   4.528,   4.823,   5.116,
     .  5.465,   5.560,   5.749,   5.347,   5.347,   5.846,   5.949,   5.982,
     .  5.976,   5.844,   5.944,   5.824,   5.485,   5.486,   5.978,   6.003,
     .  8.577,   8.481,   8.196,   7.957,   7.624,   7.169,   6.687,   6.821,
     .    6.527,    6.239,    5.999,   11.852,   11.649,   11.145,   10.692,   10.247/
      data (latitude(i),i= 3841, 3920)/
     .    9.836,    9.423,    9.052,    9.326,    9.831,   10.249,   10.733,   11.057,
     .   11.239,   11.391,   11.880,   12.111,   12.110,   11.800,   11.487,   11.449,
     .   11.193,   10.710,   10.454,   10.544,   10.618,   10.604,   10.329,   10.144,
     .   10.097,   10.262,   10.446,   10.638,   10.638,   10.769,   10.682,   10.738,
     .   10.698,   10.555,   10.055,    9.996,    9.996,    9.891,    9.783,    9.552,
     .    9.062,    8.944,    8.627,    8.813,    8.944,    8.627,    8.493,    8.549,
     .    8.578,   13.370,   13.368,   12.591,   12.590,   -2.747,   -2.746,  -55.829,
     .  -55.830,  -55.824,  -55.824,  -55.587,  -55.590,  -55.098,  -55.199,  -55.068,
     .  -55.107,  -54.966,  -55.449,  -55.230,  -55.259,  -54.991,  -54.970,  -54.945,
     .  -54.945,  -54.898,  -54.899,  -54.860,  -54.862,  -54.632,  -54.632,  -54.010/
      data (latitude(i),i= 3921, 4000)/
     .  -54.014,  -53.602,  -54.103,  -53.600,  -53.607,  -54.889,  -54.862,  -54.829,
     .  -54.683,  -54.475,  -54.487,  -54.567,  -54.216,  -54.387,  -54.466,  -54.687,
     .  -54.687,  -54.210,  -53.819,  -53.542,  -53.435,  -52.944,  -52.540,  -52.556,
     .  -52.645,  -53.942,  -53.944,  -54.312,  -54.312,  -54.079,  -54.082,  -54.076,
     .  -54.076,  -53.780,  -54.128,  -53.705,  -53.586,  -53.780,  -53.523,  -53.527,
     .  -53.397,  -53.400,  -53.252,  -53.256,  -52.727,  -52.990,  -53.275,  -53.063,
     .  -52.778,  -52.729,  -52.734,  -52.729,  -52.709,  -52.709,  -52.397,  -52.391,
     .  -52.228,  -52.224,  -51.941,  -51.943,  -52.079,  -52.073,  -51.802,  -51.802,
     .  -52.307,  -52.309,  -52.088,  -52.088,  -51.900,  -51.892,  -51.613,  -51.614,
     .  -51.473,  -51.472,  -51.207,  -51.205,  -51.055,  -51.051,  -50.847,  -50.848/
      data (latitude(i),i= 4001, 4080)/
     .  -50.889,  -50.889,  -50.794,  -50.792,  -50.434,  -50.431,  -50.050,  -49.554,
     .  -49.051,  -49.483,  -49.987,  -50.049,  -49.882,  -49.883,  -49.279,  -49.274,
     .  -49.270,  -49.270,  -52.131,  -52.130,  -49.069,  -49.068,  -48.792,  -48.792,
     .  -48.694,  -48.198,  -48.671,  -48.692,  -48.438,  -48.436,  -47.977,  -47.978,
     .  -48.693,  -48.688,  -48.706,  -48.222,  -48.697,  -48.703,  -48.040,  -48.040,
     .  -47.841,  -47.834,  -47.783,  -47.782,  -47.121,  -47.122,  -47.161,  -47.156,
     .  -46.102,  -46.097,  -45.737,  -45.730,  -45.756,  -45.757,  -45.971,  -45.972,
     .  -45.754,  -45.752,  -45.696,  -45.693,  -45.560,  -45.559,  -45.352,  -45.349,
     .  -45.274,  -45.272,  -45.398,  -45.396,  -45.300,  -45.295,  -45.025,  -45.025,
     .  -44.964,  -44.958,  -44.820,  -44.818,  -44.911,  -44.907,  -44.694,  -44.696/
      data (latitude(i),i= 4081, 4160)/
     .  -44.673,  -44.670,  -44.803,  -44.804,  -44.728,  -44.728,  -44.917,  -44.417,
     .  -44.915,  -44.918,  -44.527,  -44.526,  -44.558,  -44.556,  -44.439.  -44.433,
     .  -44.292,  -44.289,  -44.326,  -44.324,  -44.199,  -44.198,  -43.940,  -43.939,
     .  -43.635,  -43 635,  -42.998,  -42.999,  -42.639,  -42.638,  -42.556,  -42.553,
     .  -43.386,  -42.947,  -42.446,  -41.957,  -42.173,  -42.596   -43.052,  -43.380,
     .  -55.232,  -55.235,  -55.166,  -55.170,  -54.728,  -54.728,  -54.802,  -54.888,
     .  -54.787,  -54.802,  -52.646,  -53.093,  -53.579,  -53.971,  -54.263,  -54.578,
     .  -54.634,  -54.898,  -54.938,  -54.887,  -54.832,  -54.787,  -39.088,  -39.088,
     .  -27.811,  -27.811,  -26.419,  -26.419,   -0.575,   -0.575,   -0.012,   -0.012,
     .   -0.153,   -0.153,    0.201,    0.201,    0.606,    0.606,    0.645,    0.645/
      data (latitude(i),i= 4161, 4240)/
     .    0.311,    0.013,    0.309,    0.311,    0.071,    0.071,   -0.271,   -0.754,
     .   -1.186,   -1.480,   -1.733,   -1.768,   -1.655,   -1.174,   -0.670,   -0.207,
     .   -0.138,   -0.240,   -0.160,   -0.234,   -0.271,    0.873,    0.873,    2.181,
     .    2.181,    2.257,    2.257,   -1.020,   -0.576,   -1.068,   -1.409,   -1.020,
     .   -3.866,   -3.866,  -23.966,  -23.966,  -23.218,  -23.218,   12.626,   12.626,
     .   12.384,   12.062,   12.360,   12.384,   12.307,   12.307,   11.823,   11.823,
     .   11.817,   11.817,   10.966,   10.966,   11.898,   11.898,   10.806,   10.806,
     .   11.063,   11.041,   10.955,   11.063,   10.824,   10.335,   10.067,   10.039,
     .   10.274,   10.775,   10.824,   11.343,   11.343,   12.228,   12.228,   13.144.
     .   13.144,   13.718,   13.718,   13.057,   13.057,  -60.735,  -60.735,  -60.771
      data (latitude(i),i= 4241, 4320)/
     .  -60.771   -54.512,  -54.512,  -54.890,  -54.509,  -54.270,  -54.164,  -54.572.
     .  -54.890,  -56.316,  -56.316,  -56.727,  -56.727,  -57.086,  -57.086,  -57.819,
     .  -57.819,  -58.493,  -58.493,  -59.070,  -59.070,  -59.469,  -59.469,  -20.490.
     .  -20.490,  -20.494,  -20.494,    0.936,    0.936,    0.922,    0.922,  -52.022.
     .  -51.786,  -51.413,  -51.915,  -52.208,  -52.025,  -52.110,  -51.717,  -51.335,
     .  -51.379,  -51.817,  -52.028,  -52.101,  -51.900,  -51.898,  -27.187,  -27.187,
     .  -26.463,  -26.463.  -33.772,  -33.772,  -33.662,  -33.662,  -26.347,  -26.347,
     . -26.263, -26.263,  -1.307,  -1.307,  -1.344,  -1.344,  -0.691,  -0.691,
     .  -0.489,  -0.489,   0.364,   0.364,   0.648,   0.648,   0.099,  -0.358,
     .  -0.823,  -0.964,  -0.464,  -0.024,   0.099,  -0.289,  -0.289,  -0.149/
      data (latitude(i),i= 4321, 4400)/
     .  -0.149, -15.394, -15.727, -16.158, -16.235, -15.896, -15.425, -15.395,
     . -69.596, -69.262, -68.917, -68.430, -68.084, -67.888, -67.625, -67.220,
     . -67.149, -67.146, -67.059, -66.910, -66.503, -66.734, -66.162, -65.914,
     . -65.775, -66.029, -66.455, -66.354, -66.708, -66.954, -66.914, -66.821,
     . -66.630, -66.576, -66.268, -66.337, -66.503, -66.644, -66.746, -66.925,
     . -66.882, -67.219, -67.237, -70.427, -70.651, -70.544, -70.601, -70.793,
     . -71.202, -71.533, -71.951, -72.373, -72.878, -73.329, -73.448, -76.226,
     . -76.723, -77.218, -77.681, -78.174, -78.483, -78.578, -78.949, -79.014,
     . -67.625, -67.221, -67.219, -73.790, -73.666, -73.477, -73.284, -73.174,
     . -72.869, -72.459, -71.956, -71.486, -70.981, -70.489, -70.031, -69.534/
      data (latitude(i),i= 4401, 4480)/
     . -69.323, -66.288, -65.932, -65.539, -65.046, -65.120, -65.121, -64.714,
     . -64.496, -64.037, -63.867, -63.637, -63.356, -63.491, -63.675, -64.044,
     . -64.368, -64.665, -65.035, -65.427, -65.911, -65.941, -66.392, -66.256,
     . -66.169, -66.171, -66.617, -67.009, -67.466, -67.968, -68.313, -74.018,
     . -74.063, -73.583, -73.186, -72.788, -72.637, -72.256, -71.755, -71.407,
     . -70.933, -71.256, -71.750, -71.365, -70.934, -71.380, -71.420, -71.417,
     . -71.314, -71.250, -71.360, -71.364, -  .088, -70.840, -70.691, -70.605,
     . -70.400, -70.473, -70.357, -70.357, -70.698, -70.460, -70.221, -70.290,
     . -70.160, -69.971, -70.153, -70.367, -70.466, -70.509, -70.358, -70.197,
     . -70.077, -70.051, -70.050, -69.838, -69.786, -69.504, -69.116, -68.637/
      data (latitude(i),i= 4481, 4560)/
     . -68.710, -69.116, -84.890, -85.241, -85.416, -85.468, -85.107, -85.249,
     . -84.812, -84.628, -84.286, -83.806, -83.307, -82.812, -82.317, -81.885,
     . -81.389, -81.118, -81.357, -81.361, -75.705, -75.705, -75.258, -75.230,
     . -75.192, -75.035, -75.065, -75.206, -74.712, -74.348, -73.925, -73.736,
     . -73.617, -73.617, -73.356, -73.338, -72.950, -73.064, -73.044, -72.999,
     . -73.276, -73.230, -73.173, -73.243, -73.325, -73.324, -73.065, -73.158,
     . -73.041, -73.539, -73.654, -73.892, -73.485, -73.101, -73.434, -73.555,
     . -73.799, -69.322, -69.171, -68.904, -67.082, -66.604, -66.288, -68.314,
     . -68.742, -68.379, -68.874, -69.354, -69.805, -70.274, -70.719, -70.719,
     . -71.072, -71.533, -72.008, -72.234, -72.736, -73.194, -73.127, -73.565/
      data (latitude(i),i= 4561, 4640)/
     . -74.010, -74.129, -74.129, -74.571, -74.907, -75.408, -75.740, -76.047,
     . -76.246, -76.594, -76.668, -76.580, -76.360, -75.959, -75.929, -75.930,
     . -76.407, -76.852, -77.279, -77.486, -77.979, -78.004, -77.894, -77.651,
     . -78.131, -78.401, -78.819, -78.693, -78.289, -78.795, -79.175, -79.625,
     . -79.626, -79.279, -79.500,  79.980, -79.904, -80.149, -80.510, -80.894,
     . -81.118, -81.411, -81.773, -82.123, -82.454, -82.952, -83.038, -83.040,
     . -82.619, -82.212, -81.978, -81.821, -81.887, -81.487, -81.204, -80.924,
     . -80.757, -80.565, -80.565, -80.317, -79.820, -79.323, -79.335, -79.300,
     . -78.986, -78.482, -78.035, -77.621, -77.344, -76.991, -76.624, -76.403,
     . -76.295, -76.295, -76.083, -75.986, -75.925, -75.797, -75.304, -74.853/
      data (latitude(i),i= 4641, 4720)/
     . -74.501, -74.136, -74.017, -68.905, -68.425, -67.926, -67.513, -67.030,
     . -67.084, -69.201, -69.681, -69.928, -70.419, -70.331, -70.833, -71.247,
     . -71.728, -72.231, -72.371, -72.021, -71.589, -71.090, -70.680, -70.188,
     . -69.858, -69.611, -69.595, -66.734, -66.670, -66.537, -66.631, -66.632,
     . -66.591, -66.630, -66.652, -66.703, -66.479, -66.099, -65.936, -65.979,
     . -66.004, -66.004, -66.170, -66.374, -66.403, -66.719, -66.686, -66.283,
     . -66.162, -66.576, -66.748, -66.512, -66.433, -66.434, -66.905, -67.054,
     . -66.701, -66.293, -66.206, -66.133, -66.080, -65.591, -65.089, -65.370,
     . -65.871, -66.230, -66.268, -67.237, -67.626, -67.811, -67.847, -68.293,
       -68.420, -68.762, -68.768, -68.929, -68.430, -68.804, -69.169, -69.158
      data (latitude(i),i= 4721, 4800)
     . -69.305, -69.619, -69.979, -70.242, -70.427, -73.448, -73.514, -73.827,
     . -74.189, -74.673, -74.961, -75.376, -75.871, -76.226, -79.012, -79.516,
     . -79.963, -80.452, -80.669, -81.170, -81.641, -82.082, -82.550, -82.999,
     . -83.385, -83.798, -84.051, -84.306, -84.442, -84.457, -69.116, -69.542,
     . -69.317, -69.787, -69.643, -69.158, -68.725, -68.449, -68.139, -67.980,
     . -67.721, -67.475, -67.729, -67.496, -66.993, -67.057, -67.140, -67.136,
     . -66.715, -66.334, -65.994, -65.896, -65.886, -65.997, -66.396, -66.683,
     . -67.061, -67.413, -67.374, -67.531, -67.578, -67.578, -67.682, -67.778,
     . -67.859, -67.738, -68.231, -68.734, -69.201, -84.457, -84.669, -84.892,
     . -81.361, -80.909, -80.493, -80.009, -79.667, -79.422, -79.153, -78.849/
      data (latitude(i),i= 4801, 4880)/
     . -78.425, -78.158, -77.926, -77.684, -77.684, -77.185, -77.082, -77.227,
     . -77.387, -77.761, -77.454, -77.155, -76.654, -76.489, -76.414, -75.912,
     . -75.678, -75.505, -75.505, -75.490, -75.112, -75.025, -74.729, -74.745,
     . -74.732, -74.829, -74.721, -74.649, -74.649, -74.667, -74.726, -74.723,
     . -74.627, -74.344, -74.525, -74.189, -74.444, -74.856, -74.379, -74.700,
     . -75.149, -75.257, -77.131, -77.131, -77.759, -77.569, -77.345, -77.836,
     . -77.759, -73.421, -73.421, -67.448, -67.448, -66.623, -66.623, -66.263,
     . -66.263, -65.682, -65.682, -72.606, -73.102, -72.618, -72.606, -72.981,
     . -73.481, -73.290, -72.818, -72.981, -73.191, -73.191, -67.779, -67.779,
     . -67.538, -67.082, -66.661, -67.153, -67.571, -67.538, -66.302, -66.302/
      data (latitude(i),i= 4881, 4960)/
     . -65.864, -65.864, -64.883, -64.883, -64.695, -64.305, -64.806, -64.695,
     . -64.428, -64.428, -62.638, -62.462, -62.651, -62.638, -62.521, -62.521,
     . -62.361, -62.361, -62.279, -62.279, -62.196, -61.926, -62.220, -62.196,
     . -61.243, -61.243, -61.260, -61.260, -63.162, -63.162, -63.426, -63.123,
     . -63.430, -63.426, -63.583, -63.583, -64.442, -63.958, -64.072, -64.442,
     . -64.528, -64.528, -65.254, -65.254, -70.574, -70.574, -70.448, -70.448,
     . -71.041, -71.041, -70.629, -70.629, -70.162, -70.162, -70.156, -70.156,
     . -72.651, -72.594, -72.674, -72.651, -72.170, -71.938, -72.054, -71.895,
     . -72.177, -72.520, -72.459, -72.265, -72.170, -73.048, -73.048, -73.418,
     . -73.418, -73.104, -72.602, -72.824, -73.179, -73.104, -71.178, -70.857/
      data (latitude(i),i= 4961, 5040)/
     . -70.624, -71.053, -71.191, -71.178, -70.132, -70.132, -69.738, -69.738,
     . -72.684, -72.414, -71.942, -71.941, -72.067, -71.578, -71.643, -71.162,
     . -71.171, -70.695, -70.237, -69.738, -69.273, -68.882, -69.085, -69.555,
     . -70.043, -70.511, -71.007, -71.509, -72.003, -72.457, -72.634, -72.722,
     . -72.684, -63.890, -63.890, -68.736, -68.736, -69.666, -69.166, -69.651,
     . -69.666, -70.642, -70.642, -71.032, -71.032, -79.557, -79.060, -78.698,
     . -78.449, -78.902, -79.348, -79.557, -80.806, -80.429, -80.200, -80.289,
     . -80.002, -80.448, -80.806, -80.919, -80.494, -80.140, -79.698, -79.203,
     . -78.706, -78.251, -77.936, -78.088, -78.542, -79.043, -79.539, -80.040,
     . -80.467, -80.681, -80.852, -80.919, -78.658, -78.602, -78.629, -78.658/
      data (latitude(i),i= 5041, 5120)/
     . -74.858, -74.485, -74.802, -74.858, -73.137, -73.137, -62.998, -62.998,
     . -70.007, -70.007, -72.253, -72.253, -70.556, -70.556, -66.247, -66.247,
     . -65.679, -65.679, -65.465, -65.465, -66.872, -66.872, -67.625, -67.625,
     . -79.878, -79.717, -79.344, -78.852, -79.054, -79.462, -79.878, -77.126,
     . -77.126, -74.464, -74.464, -74.544, -74.544, -74.519, -74.519, -74.164,
     . -73.919, -73.643, -73.323, -73.646, -74.109, -74.164, -73.994, -74.235,
     . -74.368, -73.876, -73.781, -73.797, -73.994, -73.975, -73.975,  44.622,
     .  44.140,  43.745,  43.745,  43.391,  42.904,  42.506,  42.115,  41.981,
     .  41.981,  41.509,  41.281,  41.082,  40.988,  40.618,  40.335,  40.639,
     .  40.733,  40.733,  40.907,  40.856,  40.726,  40.245,  40.233,  40.351/
      data (latitude(i),i= 5121, 5200)/
     .  39.898,  39.881,  39.881,  39.476,  38.982,  38.547,  38.299,  37.835,
     .  37.835,  38.024,  37.539,  37.163,  36.676,  36.438,  36.386,  36.386,
     .  36.852,  36.986,  37.485,  37.829,  38.158,  38.174,  37.951,  38.378,
     .  38.373,  38.321,  38.299,  38.299,  38.750,  39.174,  39.584,  39.692,
     .  39.692,  40.124,  40.580,  41.081,  41.583,  41.849,  41.850,  42.221,
     .  42.502,  42.810,  42.976,  43.408,  43.522,  43.845,  44.234,  44.699,
     .  45.178,  44.951,  45.320,  45.596,  45.596,  45.630,  45.486,  44.994,
     .  44.511,  44.080,  44.082,  43.782,  43.452,  42.977,  42.524,  42.189,
     .  41.919,  41.934,  41.438,  41.209,  41.021,  40.739,  40.646,  40.646,
     .  40.292,  39.793,  40.174,  40.365,  40.181,  39.700,  39.431,  38.333
      data (latitude(i),i= 5201, 5280)/
     .  38.734,  38.270,  37.918,  37.918,  38.410,  38.830,  39.311,  39.776,
     .  40.074,  40.239,  40.239,  40.626,  40.959,  41.246,  41.415,  41.722,
     .  42.071,  42.089,  42.089,  42.418,  42.780,  43.242,  43.711,  44.107,
     .  44.347,  44.332,  43.926,  43.782,  43.782,  43.505,  43.170,  43.144,
     .  43.340,  43.460,  43.461,  43.157,  42.669,  42.435,  42.435,  41.934,
     .  41.635,  41.337,  41.188,  40.992,  40.577,  40.416,  40.416,  40.021,
     .  39.587,  39.086,  38.636,  38.358,  37.904,  37.606,  37.606,  37.555,
     .  37.215,  36.769,  36.692,  36.746,  36.732,  36.671,  36.671,  36.467,
     .  36.111,  36.213,  36.643,  37.080,  37.197,  37.192,  37.191,  36.999,
     .  37.121,  37.590,  38.093,  38.500,  38.971,  38.722,  38.722,  39.214/
      data (latitude(i),i= 5281, 5360)/
     .  39.670,  40.117,  40.618,  41.115,  41.602,  41.964,  41.964,  42.455,
     .  42.921,  43.303,  43.627,  43.776,  43.776,  43.596,  43.579,  43.652,
     .  43.514,  43.422,  43.401,  43.401,  43.508,  43.372,  43.360,  43.395,
     .  43.396,  43.823,  44.317,  44.818,  45.317,  45.008,  45.486,  45.707,
     .  45.707,  46.208,  46.508,  46.977,  47.021,  47.021,  47.373,  47.625,
     .  47.820,  48.004,  48.041,  48.040,  48.543,  48.713,  48.836,  48.518,
     .  48.692,  48.693,  48.640,  49.141,  49.610,  49.352,  49.341,  49.275,
     .  49.275,  49.481,  49.902,  50.210,  50.709,  51.028,  51.091,  51.091,
     .  51.346,  51.402,  51.904,  51.988,  51.988,  52.382,  52.871,  53.145,
     .  53.384,  51.651,  51.741,  51.662,  51.826,  51.686,  51.786,  53.958/
      data (latitude(i),i= 5361, 5440)/
     .  54.394,  54.669,  54.686,  55.151,  55.614,  56.118,  56.583,  56.617,
     .  56.169,  55.874,  55.432,  55.356,  56.583,  56.805,  57.052,  56.999,
     .  57.101,  57.101,  57.190,  57.571,  57.571,  57.230,  57.230,  57.078,
     .  57.078,  56.617,  55.356,  54.859,  54.669,  55.611,  55.120,  55.102,
     .  55.272,  55.767,  55.718,  55.916,  56.179,  55.734,  55.417,  55.421,
     .  54.686,  54.184,  53.814,  53.796,  53.700,  53.418,  53.414,  53.384,
     .  54.518,  54.528,  53.929,  54.171,  54.171,  54.423,  54.231,  53.941,
     .  53.958,  54.461,  54.343,  54.343,  54.721,  54.767,  54.570,  54.256,
     .  54.140,  53.947,  53.929,  65.816,  65.556,  65.059,  64.711,  64.280,
     .  63.887,  63.519,  63.416,  63.416,  63.094,  62.643,  62.154,  61.659/
      data (latitude(i),i= 5441, 5520)/
     .  61.160,  61.057,  61.057,  60.563,  60.391,  60.045,  59.926,  60.561,
     .  60.456,  60.396,  60.163,  60.004,  59.827,  59.927,  58.986,  58.492,
     .  58.079,  57.583,  57.114,  56.673,  56.179,  55.421,  55.857,  56.163,
     .  56.080,  56.538,  57.022,  57.512,  57.740,  57.740,  58.239,  58.627,
     .  58.731,  59.152,  59.303,  59.287,  59.478,  59.455,  59.581,  60.074,
     .  60.434,  60.691,  61.194,  61.313,  61.311,  61.810,  62.311,  62.759,
     .  63.171,  63.468,  63.602,  63.602,  63.955,  64.379,  64.848,  65.341,
     .  65.567,  65.806,  65.817,  69.790,  70.001,  70.257,  70.711,  70.756,
     .  70.270,  70.763,  70.434,  70.936,  70.526,  70.077,  70.579,  70.482,
     .  69.991,  70.287,  70.207,  70.207,  69.796,  69.765,  69.306,  69.791/
      data (latitude(i),i= 5521, 5600)/
     .  69.649,  69.292,  68.830,  68.481,  68.389,  68.391,  67.897,  67.874,
     .  67.783,  67.783,  67.386,  67.244,  67.204,  67.151,  66.756,  66.275,
     .  65.836,  65.332,  65.001,  64.514,  64.428,  64.428,  63.986,  63.822,
     .  63.462,  63.391,  63.074,  63.001,  62.723,  62.604,  62.103,  62.272,
     .  62.187,  62.187,  61.877,  61.682,  61.195,  61.133,  61.158,  61.044,
     .  61.054,  60.553,  60.100,  60.439,  60.003,  59.549,  59.564,  59.128,
     .  58.750,  58.750,  58.364,  58.124,  58.071,  58.359,  58.733,  59.015,
     .  59.502,  59.154,  58.987,  53.071,  52.927,  40.493,  40.493,  40.796,
     .  40.796,  40.005,  40.005,  39.205,  39.205,  38.978,  38.978,  39.039,
     .  38.704,  38.334,  38.452,  38.837,  39.039,  37.680,  37.680,  37.471/
      data (latitude(i),i= 5601, 5680)/
     .  37.471,  37.990,  37.990,  37.676,  37.676,  37.503,  37.503,  37.501,
     .  37.501,  37.174,  37.174,  36.788,  36.788,  36.477,  36.477,  35.311,
     .  35.007,  34.954,  35.095,  35.193,  35.523,  35.587,  35.418,  35.335,
     .  35.183,  35.311,  36.377,  36.377,  36.751,  36.751,  37.118,  37.118,
     .  37.043,  37.043,  37.999,  37.999,  37.769,  37.769,  37.923,  37.923,
     .  38.468,  38.468,  38.835,  38.835,  39.818,  39.818,  42.784,  42.784,
     .  42.778,  42.778,  42.985,  42.985,  43.196,  43.137,  43.188,  43.196,
     .  43.389,  43.389,  43.071,  43.071,  44.169,  44.169,  44.694,  44.344,
     .  44.691,  44.694,  45.165,  44.676,  45.157,  45.165,  45.214,  45.214,
     .  54.334,  54.334,  54.462,  54.462,  54.749,  54.749,  53.744,  53.744/
      data (latitude(i),i= 5681, 5760)/
     .  53.718,  53.718,  53.602,  53.602,  53.498,  53.498,  36.766,  36.766,
     .  38.256,  37.836,  37.358,  36.856,  36.783,  37.108,  37.298,  37.573,
     .  37.944,  38.146,  37.993,  38.012,  38.188,  38.297,  38.256,  42.844,
     .  42.844,  41.243,  40.859,  40.359,  39.857,  39.357,  39.145,  38.960,
     .  39.463,  39.964,  40.467,  40.947,  40.965,  41.243,  36.025,  36.025,
     .  35.803,  35.803,  37.001,  37.001,  37.902,  37.739,  37.902,  38.723,
     .  38.723,  38.527,  38.527,  39.519,  39.519,  40.030,  40.030,  39.966,
     .  39.472,  39.568,  39.942,  39.966,  39.123,  39.123,  38.759,  38.759,
     .  37.766,  37.766,  36.998,  36.998,  43.006,  42.509,  42.010,  41.526,
     .  41.743,  42.236,  42.634,  43.008,  43.006,  45.819,  45.819,  46.157/
      data (latitude(i),i= 5761, 5840)/
     .  46.157,  46.701,  46.701,  46.913,  46.913,  47.387,  47.387,  48.478,
     .  48.478,  49.254,  49.254,  49.500,  49.500,  52.564,  52.564,  53.182,
     .  53.182,  53.443,  53.443,  53.466,  53.466,  55.354,  55.232,  54.867,
     .  54.411,  54.041,  53.540,  53.488,  52.612,  53.081,  53.378,  53.821,
     .  54.227,  54.292,  54.471,  54.972,  55.253,  55.354,  54.010,  54.010,
     .  57.961,  57.685,  57.684,  57.191,  56.733,  56.415,  56.360,  56.360,
     .  56.051,  55.730,  55.275,  54.796,  54.675,  54.674,  54.395,  53.926,
     .  53.685,  53.684,  53.308,  52.810,  52.954,  52.689,  52.191,  51.784,
     .  51.506,  51.375,  51.347,  51.346,  50.942,  50.738,  50.797,  50.861,
     .  50.701,  50.729,  50.453,  50.362,  50.145,  50.134,  51.014,  51.241/
      data (latitude(i),i= 5841, 5920)/
     .  51.351,  51.733,  51.469,  51.609,  51.648,  52.135,  52.525,  52.798,
     .  52.789,  52.789,  53.171,  53.332,  53.727,  54.225,  54.451,  54.931,
     .  54.769,  54.770,  54.848,  55.346,  55.844,  55.489,  55.988,  56.438,
     .  56.813,  56.813,  56.563,  57.059,  57.561,  57.931,  57.971,  57.971,
     .  58.456,  58.557,  58.639,  58.176,  57.961,  53.488,  52.988,  52.503,
     .  52.353,  52.353,  52.154,  51.925,  51.643,  51.484,  51.602,  51.783,
     .  51.783,  52.143,  52.580,  52.612,  58.509,  58.012,  58.511,  58.509,
     .  57.684,  57.684,  57.488,  57.488,  57.406,  57.406,  57.049,  57.049,
     .  57.695,  57.248,  57.428,  57.695,  57.054,  57.054,  56.688,  56.688,
     .  56.534,  56.534,  56.111,  56.111,  56.142,  56.142,  55.917,  55.917/
      data (latitude(i),i= 5921, 6000)/
     .  56.650,  56.650,  51.369,  51.369,  50.134,  50.599,  51.014,  58.936,
     .  58.936,  58.850,  58.850,  50.678,  50.678,  49.934,  49.934,  53.425,
     .  53.425,  54.409,  54.409,  55.717,  55.717,  57.814,  57.814,  60.837,
     .  60.837,  60.466,  59.971,  60.466,  60.725,  60.725,  54.957,  54.957,
     .  55.272,  54.806,  55.191,  55.621,  55.964,  55.916,  55.102,  55.227,
     .  55.532,  55.611,  55.970,  55.970,  57.087,  57.087,  55.300,  55.300,
     .  55.059,  55.059,  55.148,  55.148,  55.084,  55.084,  57.303,  57.303,
     .  56.974,  56.974,  70.924,  70.924,  70.522,  70.522,  70.274,  70.274,
     .  70.478,  70.784,  70.467,  70.478,  70.246,  70.246,  69.801,  69.801,
     .  69.530,  69.530,  70.059,  70.059,  70.006,  70.006,  70.053,  70.053/
      data (latitude(i),i= 6001, 6080)/
     .  69.011,  69.501,  69.007,  69.011,  68.911,  68.911,  68.263,  68.723,
     .  68.257,  68.263,  68.617,  68.617,  68.167,  68.167,  68.044,  68.044,
     .  68.071,  68.071,  65.366,  65.366,  64.963,  64.963,  63.717,  63.717,
     .  63.551,  63.551,  63.386,  63.386,  62.334,  62.334,  62.264,  62.264,
     .  61.846,  61.846,  60.674,  60.674,  59.962,  59.962,  59.409,  59.409,
     .  58.282,  58.282,  57.338,  56.856,  56.377,  56.879,  57.337,  57.338,
     .  57.899,  57.401,  57.017,  57.507,  57.909,  57.899,  57.950,  57.950,
     .  58.390,  58.390,  64.973,  64.973,  63.244,  63.244,  60.288,  60.288,
     .  60.400,  60.400,  60.156,  60.156,  60.192,  60.192,  60.064,  60.064,
     .  66.382,  65.883,  65.523,  65.029,  64.666,  64.383,  64.383,  64.382/
      data (latitude(i),i= 6081, 6160)/
     .  63.857,  63.746,  63.409,  63.522,  63.757,  63.849,  63.849,  63.827,
     .  64.187,  64.600,  64.790,  64.890,  64.889,  64.998,  65.141,  65.535,
     .  65.444,  65.946,  65.910,  66.328,  66.219,  65.957,  65.956,  65.454,
     .  65.645,  65.990,  65.866,  66.172,  66.483,  66.271,  66.384,  61.564,
     .  61.564,  61.873,  61.873,  62.317,  62.317,  62.357,  62.357,  62.301,
     .  62.301,  62.144,  62.144,  54.286,  54.619,  54.278,  54.286,  52.927,
     .  52.424,  52.584,  53.005,  53.071,  31.239,  31.411,  31.411,  31.542,
     .  31.441,  31.441,  31.292,  31.237,  31.237,  31.095,  30.842,  30.826,
     .  27.533,  27.099,  26.614,  26.164,  25.736,  25.278,  24.808,  24.377,
     .  24.013,  23.527,  23.041,  22.686,  22.341,  22.285,  30.826,  31.049/
```

```
 data (latitude(i),i= 6161, 6240)/
.  31.098,   31.278,   31.445,   31.551,   31.570,   31.644,   27.535,   28.021,
.  28.398,   28.802,   28.802,   29.285,   29.757,   29.951,   31.644,   32.019,
.  32.000,   32.180,   32.427,   32.757,   32.912,   32.826,   32.708,   32.441,
.  32.069,   31.568,   31.210,   31.210,   30.737,   30.394,   30.273,   30.549,
.  30.838,   31.026,   31.188,   31.230,   31.388,   31.809,   32.296,   32.449,
.  32.701,   32.789,   32.849,   32.824,   33.061,   33.171,   33.171,   33.534,
.  33.648,   33.972,   34.416,   34.765,   35.196,   35.265,   35.266,   35.712,
.  36.117,   36.519,   36.980,   36.764,   37.206,   37.298,   37.120,   36.942,
.  36.942,   36.869,   37.052,   37.051,   36.893,   37.028,   36.825,   36.700,
.  36.887,   36.917,   36.787,   36.643,   36.627,   36.628,   36.549,   36.485/
 data (latitude(i),i= 6241, 6320)/
.  36.260,   35.917,   35.769,   35.613,   35.260,   35.093,   35.095,   35.095,
.  35.183,   35.199,   35.239,   35.209,   35.509,   35.831,   35.781,   35.781,
.  35.304,   34.831,   34.384,   33.963,   33.701,   33.478,   33.270,   32.871,
.  32.480,   32.044,   32.044,   31.634,   31.149,   30.645,   30.194,   29.749,
.  29.326,   28.961,   28.697,   28.325,   28.123,   27.993,   27.747,   27.665,
.  24.003,   23.559,   23.120,   22.635,   22.263,   21.816,   21.320,   20.822,
.  20.766,   27.665,   27.201,   26.728,   26.442,   26.144,   25.669,   25.196,
.  24.697,   24.347,   24.002,   20.766,   20.579,   20.201,   19.706,   19.210,
.  18.750,   18.255,   17.753,   17.265,   16.808,   16.326,   16.060,   16.061,
.  15.574,   15.155,   14.821,   14.431,   13.976,   13.530,   13.443,   13.516/
 data (latitude(i),i= 6321, 6400)/
.  13.409,   13.286,   12.851,   12.563,   12.610,   12.832,   12.553,   12.582,
.  12.323,   12.323,   12.293,   12.016,   11.921,   11.462,   11.008,   10.981,
.  10.482,   10.153,    9.872,    9.495,    9.040,    9.036,    9.035,    8.544,
.   8.112,    7.694,    7.319,    7.143,    6.856,    6.557,    6.219,    5.998,
.   5.656,    5.304,    5.003,    4.779,    4.557,    4.358,    4.353,    4.353,
.   4.536,    4.741,    4.924,    5.061,    5.119,    5.177,    5.216,    5.210,
.   5.303,    5.173,    5.352,    5.085,    5.085,    4.977,    4.781,    4.956,
.   5.116,    5.277,    5.510,    5.726,    5.784,    6.050,    6.101,    6.101,
.   6.234,    6.307,    6.363,    6.389,    6.577,    6.396,    6.226     5.875,
.   5.648,    5.151,    4.656,    4.370,    4.323,    4.759,    4.468,    4.515/
 data (latitude(i),i= 6401, 6480)/
.   4.519,    4.518,    4.543,    4.545,    4.214,    4.214,    3.901,    3.631,
.   3.144,    2.649,    2.348,    2.348,    1.850,    1.424,    1.076,    1.000,
.   1.000,    0.535,    0.185,    0.095,   -0.372,   -0.686,   -0.686,   -1.170,
.  -1.571,   -1.609,   -2.327,   -2.530,   -2.968,   -3.331,   -3.703,   -3.944,
.  -1.613,   -1.389,   -1.846,   -2.276,   -2.326,   -3.946,   -4.304,   -4.708,
.  -5.019,   -5.858,   -5.970,   -5.770,   -5.266,   -5.019,  -12.772,  -13.209,
. -13.665,  -14.139,  -14.637,  -15.114,  -15.609,  -16.070,  -16.570,  -17.072,
. -17.207,   -5.858,   -6.011,   -6.448,   -6.861,   -7.333,   -7.803,   -8.261,
.  -8.762,   -9.096,   -9.336,   -9.337,   -9.821,  -10.272,  -10.697,  -11.198,
. -11.700,  -12.185,  -12.599,  -12.773,  -17.207,  -17.709,  -18.187,  -18.615/
 data (latitude(i).i= 6481, 6560)/
. -19.021,  -19.464,  -19.909,  -20.357,  -20.837,  -21.259,  -21.694,  -22.100,
. -22.548,  -22.682,  -22.682,  -23.138,  -23.138,  -23.637,  -24.137,  -24.616,
. -25.081,  -25.582,  -26.076,  -26.556,  -27.055,  -27.522,  -27.970,  -28.327,
. -28.577,  -28.577,  -29.027,  -29.484,  -29.965,  -30.439,  -30.892,  -31.327,
. -31.742,  -32.236,  -32.722,  -33.214,  -33.640,  -34.138,  -34.348,  -34.348,
. -34.334,  -34.692,  -34.698,  -34.441,  -34.428,  -34.351,  -34.063,  -34.085,
. -33.992,  -34.063,  -34.187,  -33.998,  -34.025,  -34.026,  -33.739,  -33.639,
. -33.386,  -33.090,  -32.759,  -32.421,  -32.056,  -31.668,  -31.355,  -30.962,
. -30.534,  -30.086,  -29.882,  -29.880,  -29.423,  -29.043,  -28.753,  -28.330,
. -27.844,  -27.358,  -26.871,  -26.847,  -26.849,  -26.347,  -25.979,  -25.565/
 data (latitude(i),i= 6561, 6640)/
. -25.262,  -25.080,  -24.906,  -24.856,  -24.855,  -24.661,  -24.290,  -23.813,
. -23.312,  -22.809,  -22.305,  -21.853,  -21.378,  -20.969,  -20.969,  -20.582,
. -20.081,  -19.604,  -19.720,  -19.368,  -19.002,  -18.863,  -14.879,  -14.380,
. -13.877,  -13.375,  -12.884,  -12.388,  -11.888,  -11.388,  -10.892,  -10.478,
. -18.862,  -18.473,  -18.099,  -17.701,  -17.463,  -17.246,  -17.065,  -16.831,
. -16.568,  -16.199,  -15.776,  -15.372,  -14.867,  -14.879,  -10.479,  -10.145,
.  -9.752,   -9.256,   -8.782,   -8.304,   -7.826,   -7.329,   -6.826.   -6.456,
.  -5.992,   -5.526,   -5.032,   -4.671,   -4.671,   -4.317,   -3.866,   -3.383,
     .  -2.923,  -2.535,  -2.226,  -1.958,  -1.671,  -1.674,  -1.256,  -0.833,
     .  -0.467,  -0.105,   0.002,   0.001,   0.368,   0.729,   1.046,   1.389/
      data (latitude(i),i= 6641, 6720)/
     .   1.681,   1.940,   2.187,   2.450,   2.811,   3.155,   3.527,   3.884,
     .   3.937,   3.937,   4.317,   4.730,   5.149,   5.558,   5.988,   6.468,
     .   6.920,   7.313,   7.313,   7.760,   8.176,   8.629,   9.046,   9.506,
     .  10.003,  10.423,  10.452,  10.452,  10.869,  11.373,  11.871,  11.948,
     .  11.592,  11.453,  11.328,  11.244,  11.244,  11.288,  11.126,  11.181,
     .  10.932,  10.713,  10.791,  10.692,  10.469,  10.416,  10.621,  10.961,
     .  11.384,  11.574,  12.708,  12.276,  11.853,  11.567,  11.574,  18.033,
     .  17.574,  17.106,  16.621,  12.708,  13.032,  13.369,  13.762,  14.068,
     .  14.477,  14.536,  14.536,  14.704,  14.913,  15.364,  15.534,  15.959,
     .  16.451,  16.618,  18.033,  18.272,  18.651,  19.001,  19.493,  19.988/
      data (latitude(i),i= 6721, 6800)/
     .  20.489,  20.987,  21.474,  21.964,  22.287,  31.138,  31.137,  31.216,
     .  31.217,  34.739,  34.739,  33.798,  33.798,  37.532,  37.532,  35.924,
     .  35.924,  29.143,  29.143,  28.500,  28.094,  28.294,  28.500,  27.939,
     .  27.939,  28.562,  28.364,  28.480,  28.562,  28.200,  28.200,  27.756,
     .  27.756,  28.529,  28.529,  30.034,  30.034,  32.822,  32.822,  33.061,
     .  33.061,  14.899,  14.899,  15.001,  15.001,  15.001,  15.001,  15.293,
     .  15.293,  16.002,  16.002,  16.793,  16.793,  16.656,  16.656,  16.803,
     .  16.803,  17.003,  17.003,  19.853,  19.853,  11.878,  11.878,  11.626,
     .  11.626,  11.601,  11.601,  11.565,  11.565,  11.251,  11.251,  11.193,
     .  11.193,  11.202,  11.202,  11.195,  11.195,  11.302,  11.302,   7.614/
      data (latitude(i),i= 6801, 6880)/
     .   7.614,  -1.421,  -1.421,   0.404,   0.404,   1.690,   1.690,   3.748,
     .   3.274,   3.751,   3.748, -16.792, -16.792, -26.014, -26.014, -21.714,
     . -21.714, -11.379, -11.846, -11.386, -11.379, -12.243, -12.243, -12.162,
     . -12.162, -12.675, -12.675, -12.748, -12.748,  -9.339,  -9.339,  -9.343,
     .  -9.343,  -7.978,  -7.978,  -5.939,  -6.385,  -5.887,  -5.939,  -5.438,
     .  -4.936,  -5.430,  -5.438,  -4.073,  -4.073,  -2.202,  -2.202,  15.605,
     .  15.605, -20.998, -20.998,  -4.281,  -4.281,  -4.551,  -4.551,  -7.115,
     .  -7.115, -10.330, -10.330, -10.397, -10.397, -20.123, -20.438, -20.123,
     . -19.677, -19.677, -13.400, -13.400, -17.074, -17.074, -25.522, -25.499,
     . -25.289, -25.023, -24.553, -24.082, -23.582, -23.099, -22.669, -22.165/
      data (latitude(i),i= 6881, 6960)/
     . -21.671, -21.274, -20.799, -20.388, -20.383, -20.383, -19.932, -19.429,
     . -18.974, -18.480, -17.980, -17.479, -17.047, -16.588, -16.198, -15.956,
     . -15.782, -15.712, -15.419, -15.456, -14.961, -14.852, -20.997, -21.489,
     . -21.969, -22.444, -22.945, -23.433, -23.925, -24.391, -24.861, -25.173,
     . -25.245, -25.509, -25.522, -14.850, -14.627, -14.162, -13.665, -13.579,
     . -13.280, -12.789, -12.298, -11.948, -11.948, -12.384, -12.828, -13.299,
     . -13.774, -14.274, -14.774, -15.214, -15.707, -15.989, -15.989, -15.548,
     . -16.010, -16.483, -16.919, -17.401, -17.904, -18.404, -18.869, -19.346,
     . -19.825, -20.298, -20.778, -20.997, -46.451, -46.451, -46.120, -46.120,
     . -46.958, -46.958, -40.391, -40.391, -37.115, -37.115, -15.994, -15.994/
      data (latitude(i),i= 6961, 7040)/
     .  -7.983,  -7.983,   1.009,   1.511,   1.855,   2.164,   1.662,   1.362,
     .   1.009,  -4.309,  -3.814,  -3.342,  -3.838,  -4.298,  -4.798,  -5.256,
     .  -5.745,  -5.922,  -5.921,  -6.417,  -6.630,  -7.071,  -7.570,  -7.997,
     .  -8.426,  -8.489,  -8.021,  -7.889,  -7.890,  -7.398,  -7.064,  -6.670,
     .  -6.199,  -5.697,  -5.234,  -4.735,  -4.250,  -4.307, -17.974, -17.566,
     . -17.117, -16.791, -16.557, -16.559, -16.780, -17.068, -17.351, -17.709,
     . -17.974,  -9.700, -10.180, -10.666, -11.169, -11.664, -12.089, -12.586,
     . -13.070, -13.563, -14.051, -14.175, -13.703, -13.200, -12.697, -12.206,
     . -11.760, -11.566, -11.565, -11.156, -10.655, -10.165,  -9.723,  -9.700,
     .  -2.046,  -1.563,  -1.071,  -0.569,  -0.100,   0.045,   0.046,   0.184
      data (latitude(i),i= 7041, 7120)/
     .   0.339,   0.325,  -0.070,  -0.190,  -0.191,  -0.092,  -0.527,  -0.937,
     .  -1.327,  -1.762,  -2.109,  -2.112,  -2.533,  -2.557,  -2.557,  -2.393,
     .  -2.882,  -2.409,  -2.389,  -2.781,  -2.326,  -2.044,  11.632,  12.026,
     .  12.296,  11.790,  11.632, -17.095, -17.095,  35.926,  36.426,  36.910,
     .  36.587,  36.718,  36.641,  36.251,  36.138,  36.021,  36.021,  36.284,
     .  36.599,  36.819,  36.825,  36.327,  36.222,  36.263,  36.710,  36.697,
      .  36.697,  36.567,  36.662,  36.788,  37.007,  37.476,  37.978,  38.204,
      .  38.638,  38.373,  38.866,  39.269,  39.471,  39.470,  39.966,  40.316,
      .  40.464,  40.507,  40.380,  40.646,  40.719,  40.900,  41.156,  41.212,
      .  41.212,  41.075,  41.340,  41.582,  41.833,  42.002,  41.979,  41.940/
       data (latitude(i),i= 7121, 7200)/
      .  42.091,  42.090,  41.684,  41.655,  41.241,  41.150,  41.118,  40.920,
      .  41.038,  41.008,  40.976,  35.005,  35.502,  35.926,  35.005,  34.648,
      .  34.648,  34.224,  33.752,  33.291,  33.095,  33.095,  32.620,  32.132,
      .  31.673,  31.596,  31.596,  31.326,  31.326,  31.122,  31.179,  31.044,
      .  31.044,  31.193,  29.951,  29.466,  29.069,  28.586,  28.294,  28.294,
      .  27.913,  28.119,  28.542,  28.542,  29.022,  29.486,  29.552,  29.552,
      .  29.546,  29.546,  29.360,  29.360,  28.870,  28.378,  28.139,  28.139,
      .  27.964,  27.964,  27.524,  27.096,  26.693,  26.251,  25.805,  25.441,
      .  24.976,  24.486,  24.174,  24.174,  23.896,  23.499,  23.018,  22.583,
      .  22.083,  21.583,  21.091,  20.685,  20.569,  20.568,  20.274,  20.000/
       data (latitude(i),i= 7201, 7280)/
      .  19.611,  19.167,  18.688,  18.270,  17.838,  17.531,  17.050,  16.686,
      .  16.196,  15.706,  15.214,  14.735,  14.244,  13.773,  13.279,  13.278,
      .  12.837,  12.603,  12.694,  12.821,  13.057,  13.369,  13.417,  13.479,
      .  13.631,  13.912,  13.989,  14.098,  14.497,  14.512,  14.512,  14.750,
      .  14.843,  15.055,  15.165,  15.325,  15.578,  16.071,  16.448,  16.628,
      .  16.749,  16.977,  17.036,  17.034,  17.174,  17.174,  17.661,  17.904,
      .  17.937,  18.359,  18.788,  18.935,  19.309,  19.810,  20.004,  20.004,
      .  20.460,  20.391,  24.748,  25.181,  25.599,  26.048,  26.550,  26.964,
      .  27.331,  27.609,  28.078,  28.545,  28.546,  20.391,  20.844,  21.245,
      .  21.585,  22.033,  22.522,  22.765,  23.168,  23.566,  23.705,  23.797/
       data (latitude(i),i= 7281, 7360)/
      .  23.988,  24.372,  24.804,  25.303,  25.636,  25.636,  26.128,  26.066,
      .  26.066,  25.656,  25.312,  24.948,  24.546,  24.310,  24.052,  24.124,
      .  24.122,  23.984,  24.353,  24.617,  24.618,  24.986,  25.480,  25.972,
      .  25.731,  25.243,  24.747,  28.545,  28.997,  29.386,  29.864,  30.012,
      .  30.012,  29.923,  29.964,  29.963,  30.371,  30.165,  30.216,  29.791,
      .  29.409,  28.947,  28.476,  28.154,  35.672,  35.373,  34.974,  34.699,
      .  34.673,  35.174,  35.361,  35.390,  35.574,  35.672,  40.145,  40.145,
      .  39.326,  39.326,  38.538,  38.538,  37.679,  37.679,  37.771,  37.771,
      .  37.004,  37.004,  36.859,  36.859,  36.636,  36.636,  36.443,  35.999,
      .  36.443,  35.815,  35.815,  17.003,  17.003,  16.876,  16.584,  16.876/
       data (latitude(i),i= 7361, 7440)/
      .  12.240,  12.240,  12.651,  12.321,  12.504,  12.624,  12.651,  17.558,
      .  17.558,  20.692,  20.218,  20.693,  20.692,  24.230,  24.230,  24.545,
      .  24.545,  24.589,  24.589,  24.326,  24.326,  26.240,  26.240,  29.983,
      .  29.983,  30.778,  30.778,  30.758,  30.758,  30.900,  30.900,  30.753,
      .  30.752,  30.823,  30.823,  30.811,  30.811,  30.593,  30.593,  30.419,
      .  30.419,  30.678,  30.745,  30.701,  30.678,  31.194,  30.812,  30.812,
      .  30.588,  30.564,  30.420,  30.176,  29.952, -37.895, -37.895, -38.722,
      . -38.722, -49.719, -49.561, -49.228, -48.738, -48.983, -49.055, -49.410,
      . -49.719, -49.083, -49.083, -52.988, -52.988, -54.459, -54.459,  53.003,
      .  53.003,  52.446,  52.446,  52.119,  52.119,  51.649,  51.408,  51.618/
       data (latitude(i),i= 7441, 7520)/
      .  51.649,  51.829,  51.829,  51.909,  51.909,  51.889,  51.889,  51.972,
      .  51.972,  51.857,  51.857,  51.876,  51.876,  51.909,  51.909,  52.095,
      .  52.095,  51.987,  51.987,  51.970,  51.970,  52.419,  52.048,  52.419,
      .  52.123,  52.106,  52.095,  52.123,  52.382,  52.382,  52.524,  52.524,
      .  52.684,  52.684,  52.778,  52.778,  53.035,  53.035,  52.855,  52.855,
      .  53.496,  53.131,  52.833,  53.267,  53.518,  53.496,  53.967,  53.525,
      .  53.324,  53.676,  53.967,  54.117,  54.117,  54.087,  54.087,  54.125,
      .  54.125,  54.292,  54.292,  56.613,  56.613,  57.209,  57.209,  60.410,
      .  59.976,  59.852,  60.161,  60.380,  60.410,  60.586,  60.586,  62.943,
      .  63.330,  63.327,  63.667,  63.436,  62.943,  65.749,  65.749,  52.022
       data (latitude(i),i= 7521, 7600)/
      .  52.022,  20.272,  20.060,  19.692,  19.279,  18.927,  19.381,  19.881,
      .  20.272,  20.939,  20.939,  20.606,  20.606,  20.925,  20.925,  21.203,
      .  21.203,  21.697,  21.697,  22.183,  22.183,  22.009,  22.009,  21.641,
      .  21.641,  23.043,  23.043,  23.578,  23.578,  25.011,  25.011,  25.788,
     .  25.788.  26.078,  26.078,  28.209,  28.209,  28.219,  28.219,  28.391,
     .  28.391,  16.729,  16.729,  44.623,  44.814,  45.211,  71.141,  71.141,
     .  74.512,  74.512,  76.684,  76.684,  78.001,  77.533,  77.487,  77.984,
     .  78.001,  78.243,  78.243,  78.804,  78.804,  78.960,  78.960,  80.342,
     .  80.036,  79.592,  79.206,  79.381,  79.711,  80.129,  80.421,  80.038,
     .  80.299,  80.342,  80.221,  80.221,  79.538,  79.081,  78.589,  78.201/
      data (latitude(i),i= 7601, 7680)/
     .  78.683,  78.409,  78.104,  77.926,  77.676,  77.186   77.001,  77.440,
     .  77.916,  78.400,  78.668,  79.095,  79.551,  79.959,  79.502,  79.010,
     .  79.477,  79.549,  79.538,  54.286,  54.619,  54.278,  54.286,  27.140,
     .  26.695,  26.205,  25.756,  25.679,  25.593,  25.396,  25.398,  25.332,
     .  25.262,  25.117,  25.197,  25.198,  25.216,  25.259,  25.230,  25.373,
     .  25.281,  25.323,  25.383,  25.423,  25.593,  25.593,  25.172,  24.769,
     .  24.343,  24.064,  24.064,  23.928,  23.928,  23.910,  23.910,  23.937,
     .  23.938,  23.774,  23.774,  23.844,  23.430,  23.064,  22.825,  22.854,
     .  22.936,  22.550,  22.373,  22.274,  22.275,  21.803,  21.464,  21.072,
     .  20.787,  20.760,  20.968,  21.126,  21.127,  21.523,  22.001,  22.277/
      data (latitude(i),i= 7681, 7760)/
     .  21.772,  21.747,  21.494,  21.033,  20.529,  20.057,  19.554,  19.217,
     .  19.217,  18.750,  18.273,  17.780,  17.295,  16.798,  16.303,  15.854,
     .  15.428,  14.945,  14.518,  14.244,  14.244,  13.754,  13.261,  12.771,
     .  12.313,  11.921,  11.469,  11.016,  10.543,  10.070,   9.666,   9.676,
     .   9.658,   9.675,   9.666,   9.658,   9.173,   8.744,   8.354,   8.087,
     .   8.288,   8.656,   8.655,   9.082,   9.247,   9.199,   9.435,   9.887,
     .  10.152,  10.152,  10.261,  10.752,  11.255,  11.753,  12.214,  12.662,
     .  13.156,  13.593,  14.088,  14.562,  14.562,  15.054,  15.531,  15.875,
     .  15.941,  16.369,  16.386,  16.680,  17.169,  17.412,  17.708,  18.083,
     .  18.333,  18.332,  18.708,  19.144,  19.454,  19.883,  19.820,  20.044/
      data (latitude(i),i= 7761, 7840)/
     .  20.407,  20.901,  21.400,  21.611,  21.829,  22.329,  22.235,  22.236,
     .  21.744,  22.062,  21.566,  21.918,  21.918,  22.229,  21.729,  22.133,
     .  22.471,  22.614,  22.128,  22.086,  22.584,  23.088,  23.460,  23.460,
     .  23.100,  22.655,  22.753,  22.365,  21.883,  21.388,  20.930,  21.055,
     .  21.055,  20.605,  20.180,  19.816,  19.910,  19.459,  19.014,  19.364,
     .  19.385,  19.198,  19.255,  19.385,  18.882,  18.447,  17.964,  17.472,
     .  16.982,  16.514,  16.016,  16.338,  16.161,  16.160,  15.788,  15.928,
     .  16.267,  16.746,  16.619,  17.077,  17.451,  17.451,  17.101,  16.606,
     .  16.187,  15.691,  15.198,  14.726,  14.228,  13.727,  13.382,  13.319,
     .  13.320,  12.843,  12.347,  11.846,  11.347,  10.863,  10.365,  10.349/
      data (latitude(i),i= 7841, 7920)/
     .  10.349,   9.878,   9.402,   8.911,   8.412,   8.225,   7.868,   7.454,
     .   7.130,   6.664,   6.419,   6.237,   6.576,   6.924,   6.959,   7.199,
     .   7.634,   7.255,   7.749,   8.242,   8.631,   9.129,   9.190,   9.189,
     .   9.189,   9.653,  10.156,  10.639,  11.109,  11.577,  12.024,  12.521,
     .  13.015,  13.364,  13.372,  13.500,  13.238,  12.745,  12.657,  12.657,
     .  12.705,  12.388,  12.065,  11.709,  11.641,   6.420,   5.969,   5.468,
     .   4.969,   4.511,   4.031,   3.609,   3.183,   2.708,   2.407,   2.158,
     .   2.039,   2.039,   1.737,   1.499,   1.435,   1.886,   2.348,   2.717,
     .   3.200,   3.505,   3.504,   4.004,   4.507,   4.994,   5.429,   5.725,
     .   6.132,   6.237,  11.641,  11.174,  11.086,  10.589,  10.557,  10.423/
      data (latitude(i),i= 7921, 8000)/
     .  10.423,  10.129,   9.626,   9.187,   9.187,   8.709,   8.961,   9.234,
     .   9.412,   9.412,   9.870,   9.565,  10.058,  10.088,  10.482,  10.481,
     .  10.489,  10.576,  10.807,  11.060,  11.314,  11.715,  12.215,  12.714,
     .  13.079,  13.078,  13.575,  14.078,  14.559,  15.037,  15.486,  15.893,
     .  16.284,  16.539,  16.888,  17.013,  17.012,  17.360,  17.740,  18.218,
     .  18.554,  19.000,  19.467,  19.942,  20.203,  20.302,  20.302,  20.724,
     .  20.995,  21.336,  21.545,  22.510,  22.549,  21.545,  21.607,  21.642,
     .  21.744,  21.452,  21.453,  21.015,  20.527,  20.257,  20.756,  21.251,
     .  21.401,  21.481,  21.481,  21.616,  21.837,  21.889,  22.049,  22.547,
     .  22.188,  22.686,  23.154,  22.662,  22.521,  22.520,  22.511,  22.549
      data (latitude(i),i= 8001, 8080)/
     .  22.449,  22.450,  22.754,  22.686,  22.833,  23.002,  23.415,  23.710,
     .  24.021,  24.460,  24.461,  24.584,  24.869,  25.185,  25.568,  25.967,
     .  26.283,  26.776,  26.626,  27.110,  27.451,  27.924,  28.110,  28.109,
     .  28.233,  28.591,  29.088,  29.523,  29.978,  30.305,  30.123,  30.195,
     .  30.195,  30.390,  30.650,  30.863,  31.352,  31.672,  31.992,  31.918,
     .  32.274,  31.943,  32.069,  31.854,  31.689,  32.116,  32.510,  32.982,
     .  33.440,  33.903,  34.354,  34.566,  40.097,  39.635,  39.539,  39.037,
     .  38.639,  38.589,  38.139,  37.899,  37.844,  37.798,  37.834,  38.619,
     .  38.959,  39.263,  39.750,  40.030,  40.290,  40.592,  40.863,  41.361,
     .  41.858,  42.219,  42.291,  37.834,  37.331,  36.844,  36.866,  36.430/
      data (latitude(i),i= 8081, 8160)/
     .  36.061,  36.058,  35.652,  35.157,  34.674,  34.684,  34.776,  34.949,
     .  35.117,  35.116,  35.117,  35.455,  35.934,  36.429,  36.932,  37.410,
     .  37.840,  38.272,  38.619,  42.289,  42.596,  43.037,  43.219,  42.853,
     .  42.757,  42.935,  43.231,  43.503,  43.947,  44.364,  44.742,  45.120,
     .  45.491,  45.877,  46.303,  46.764,  47.198,  47.630,  48.035,  48.415,
     .  48.566,  48.567,  49.056,  49.543,  50.041,  50.527,  51.010,  51.493,
     .  51.907,  52.392,  52.893,  53.254,  53.254,  53.569,  53.984,  54.227,
     .  54.296,  53.792,  53.577,  54.014,  53.798,  54.297,  54.599,  54.739,
     .  55.151,  55.524,  55.811,  56.156,  56.600,  56.996,  57.318,  57.703,
     .  58.097,  58.485,  58.865,  59.229,  59.341,  59.375,  59.410,  59.456/
      data (latitude(i),i= 8161, 8240)/
     .  59.285,  59.396,  59.688,  59.652,  59.582,  59.271,  58.903,  59.011,
     .  59.092,  59.259,  59.267,  59.137,  59.534,  59.990,  60.408,  60.718,
     .  61.167,  61.589,  61.773,  61.903,  61.847,  61.344,  60.866,  61.054,
     .  61.429,  61.714,  62.198,  62.609,  62.567,  62.450,  62.452,  62.463,
     .  62.010,  61.524,  61.042,  60.678,  60.423,  59.988,  59.599,  59.244,
     .  58.804,  58.408,  58.072,  57.817,  57.402,  56.975,  56.575,  56.084,
     .  55.584,  55.079,  54.582,  54.091,  53.596,  53.098,  52.603,  52.118,
     .  51.618,  51.132,  51.352,  51.717,  52.154,  52.652,  52.651,  53.044,
     .  53.240,  53.740,  54.215,  54.538,  54.516,  54.971,  55.441,  55.921,
     .  56.328,  56.641,  57.095,  57.569,  57.802,  58.288,  58.733,  59.162/
      data (latitude(i),i= 8241, 8320)/
     .  59.629,  59.824,  59.824,  59.987,  60.132,  60.433,  59.932,  60.245,
     .  60.537,  60.567,  60.381,  59.941,  60.394,  60.714,  61.066,  61.458,
     .  61.701,  61.884,  62.174,  62.434,  62.863,  62.866,  62.561,  62.371,
     .  62.741,  63.222,  63.519,  64.023,  64.372,  64.663,  64.746,  64.845,
     .  64.704,  64.704,  65.030,  65.072,  65.505,  65.973,  66.410,  65.927,
     .  65.478,  65.599,  65.437,  64.948,  64.733,  64.386,  64.619,  64.616,
     .  64.425,  64.846,  65.267,  65.514,  65.898,  66.399,  66.781,  67.001,
     .  67.086,  66.603,  67.030,  67.517,  67.895,  68.184,  68.522,  68.821,
     .  68.982,  69.296,  69.538,  69.771,  69.852,  69.839,  69.842,  69.971,
     .  70.094,  69.666,  69.240,  68.749,  69.011,  69.285,  69.768,  69.522/
      data (latitude(i),i= 8321, 8400)/
     .  69.597,  69.691,  69.646,  69.241,  68.743,  69.236,  69.678,  70.133,
     .  70.637,  70.959,  71.085,  70.98[illegible],  70.915,  70.789,  70.989,  70.987,
     .  71.394,  71.669,  72.173,  72.329,  72.026,  71.677,  72.171,  72.269,
     .  72.345,  72.636,  72.702,  72.863,  72.475,  71.983,  71.490,  71.548,
     .  71.472,  71.513,  71.515,  71.505,  71.507,  71.289,  70.817,  71.080,
     .  71.506,  72.008,  72.309,  71.832,  72.279,  72.542,  72.726,  72.920,
     .  72.999,  73.181,  73.581,  73.695,  73.580,  73.091,  72.607,  72.595,
     .  72.595,  73.071,  73.569,  73.765,  74.015,  73.508,  73.307,  73.144,
     .  72.802,  73.281,  73.622,  73.948,  74.345,  74.680,  74.996,  75.459,
     .  75.836,  76.328,  76.700,  76.734,  76.518,  77.018,  77.084,  77.551/
      data (latitude(i),i= 8401, 8480)/
     .  77.630,  77.291,  76.840,  76.419,  76.479,  75.975,  75.474,  75.166,
     .  75.170,  75.592,  76.078,  76.057,  75.861,  76.115,  75.758,  75.608,
     .  75.350,  75.151,  74.645,  74.146,  73.670,  73.383,  72.994,  72.991,
     .  73.406,  73.778,  73.699,  73.658,  73.641,  73.339,  72.859,  72.364,
     .  72.040,  71.691,  71.203,  70.720,  70.254,  70.086,  70.086,  70.589,
     .  71.007,  71.433,  71.702,  72.104,  72.381,  72.210,  71.979,  71.492,
     .  71.275,  71.070,  71.070,  71.302,  71.795,  72.269,  72.765,  72.274,
     .  71.909,  71.482,  71.022,  70.556,  70.092,  69.593,  69.091,  69.189,
     .  69.136,  68.796,  68.314,  67.871,  67.601,  67.567,  67.567,  67.555,
     .  68.020,  68.523,  68.956,  68.867,  68.466,  67.980,  67.513,  67.039/
      data (latitude(i),i= 8481, 8560)/
     .  66.682,  66.281,  66.367,  66.474,  66.870,  66.870,  66.520,  66.973,
     .  67.312,  67.724,  68.228,  68.729,  69.176,  69.680,  70.184,  70.687,
     .  71.181,  71.676,  72.158,  72.662,  72.909,  72.908,  72.888,  72.476,
     .  71.986,  71.521,  71.177,  70.705,  70.205,  69.714,  69.509,  69.020,
     .  68.538,  68.312,  68.312,  68.609,  68.935,  69.206,  69.533,  69.717,
     .  69.794,  69.478,  69.035,  68.707,  68.974,  68.696,  68.582,  68.581,
     .  68.486,  68.219,  68.721,  68.665,  68.396,  68.142,  67.855,  67.618,
     .  67.118,  66.823,  66.824,  66.825,  67.239,  67.739,  67.813,  68.273,
     .  68.578,  68.632,  68.168,  67.666,  67.177,  66.728,  66.242,  66.411,
     .  66.234,  66.167,  66.167,  65.902,  65.543,  65.065,  64.588,  64.627/
      data (latitude(i),i= 8561, 8640)/
     .  64.718,  65.124,  64.831,  64.409,  64.106,  63.924,  63.924,  63.967,
     .  64.346,  64.684,  65.174,  65.680,  66.160,  66.480,  66.872,  67.151,
     .  67.150,  66.912,  66.725,  66.461,  66.297,  66.129,  66.078,  66.266,
     .  66.641,  67.101,  67.586,  67.958,  68.059,  68.059,  68.369,  68.731,
     .  69.035,  69.271,  69.309,  69.457,  69.947,  69.790,  60.561,  60.337,
     .  60.189,  59.847,  59.763,  59.445,  59.577,  59.490,  59.490,  59.473,
     .  59.219,  58.730,  58.247,  57.753,  57.737,  57.743,  57.243,  56.973,
     .  57.377,  57.700,  57.429,  56.954,  56.467,  55.965,  55.473,  54.971,
     .  55.467,  55.042,  54.741,  54.462,  40.729,  40.891,  40.901,  38.289,
     .  38.122,  37.955,  37.451,  37.123,  37.151,  40.731,  40.228,  39.879/
      data (latitude(i),i= 8641, 8720)/
     .  39.386,  38.934,  38.943,  39.324,  39.594,  39.823,  39.833,  40.096,
     .  38.289,  38.764,  39.199,  39.155,  39.329,  39.782,  40.052,  40.327,
     .  40.699,  40.901,  34.566,  34.943,  35.363,  35.753,  36.212,  36.433,
     .  36.697,  36.898,  36.922,  37.395,  37.544,  37.483,  37.771,  37.683,
     .  37.340,  37.151,  45.212,  45.714,  45.317,  45.580,  45.817,  45.817,
     .  46.206,  46.607,  46.625,  46.625,  46.720,  47.209,  46.731,  46.237,
     .  46.129,  46.041,  45.683,  45.194,  44.760,  44.395,  44.743,  44.908,
     .  44.991,  45.284,  45.385,  45.592,  46.055,  46.499,  46.256,  46.256,
     .  46.611,  46.750,  46.924,  47.097,  47.238,  47.116,  46.861,  46.622,
     .  46.671,  46.671,  46.300,  45.814,  45.433,  45.424,  44.982,  44.637/
      data (latitude(i),i= 8721, 8800)/
     .  44.368,  44.155,  43.797,  43.431,  43.318,  43.318,  43.074,  42.760,
     .  42.298,  41.809,  41.388,  41.173,  40.940,  40.976,  24.073,  24.073,
     .  21.942,  21.942,  21.867,  21.866,  21.886,  21.886,  21.595,  21.594,
     .  -0.603,  -0.603,  -0.656,  -0.656,  -0.682,  -0.682,  -0.671,  -0.671,
     .  -0.626,  -0.626,  -0.589,  -0.589,  -0.284,  -0.284,   0.496,   0.496,
     .   0.403,   0.403,   0.305,   0.305,   0.270,   0.270,   0.228,   0.228,
     .   0.240,   0.240,   0.285,   0.285,   0.388,   0.388,   0.508,   0.508,
     .   0.848,   0.848,   0.801,   0.801,   0.843,   0.843,   2.033,   2.033,
     .   1.991,   1.991,   1.893,   1.893,   1.830,   1.830,   1.791,   1.791,
     .   1.806,   1.806,   1.843,   1.843,   1.933,   1.933,   1.976,   1.976/
      data (latitude(i),i= 8801, 8880)/
     .   2.027,   2.027,   2.082,   2.082,   2.110,   2.110,   2.327,   2.327,
     .   2.293,   2.293,   2.187,   2.187,   2.200,   2.200,   2.348,   2.348,
     .   2.437,   2.437,   2.506,   2.506,   2.694,   2.694,   2.682,   2.682,
     .   2.742,   2.742,   2.953,   2.953,   2.778,   2.778,   2.904,   2.904,
     .   2.893,   2.893,   2.916,   2.916,   2.955,   2.955,   2.962,   2.962,
     .   3.110,   3.110,   3.374,   3.374,   3.451,   3.451,   3.552,   3.552,
     .   3.511,   3.511,   3.517,   3.517,   3.727,   3.727,   4.003,   4.003,
     .   4.266,   4.266,   3.945,   3.945,   3.847,   3.847,   3.942,   3.942,
     .   3.973,   3.973,   4.104,   4.104,   4.173,   4.173,   4.178,   4.178,
     .   4.191,   4.191,   4.342,   4.342,   4.371,   4.371,   4.435,   4.435/
      data (latitude(i),i= 8881, 8960)/
     .   4.453,   4.453,   4.474,   4.474,   4.630,   4.630,   4.896,   4.896,
     .   4.971,   4.971,   5.133,   5.133,   5.173,   5.173,   5.299,   5.299,
     .   5.454,   5.454,   5.368,   5.368,   5.362,   5.362,   5.308,   5.308,
     .   5.283,   5.283,   5.419,   5.419,   5.981,   5.981,   5.969,   5.969,
     .   5.864,   5.864,   5.814,   5.814,   5.771,   5.771,   5.738,   5.738,
     .   5.841,   5.841,   5.911,   5.911,   5.970,   5.970,   6.157,   6.157,
     .   6.189,   6.189,   6.240,   6.240,   6.287,   6.287,   6.330,   6.330,
     .   6.327,   6.327,   6.351,   6.351,   6.442,   6.442,   6.412,   6.412,
     .   6.622,   6.622,   6.668,   6.668,   6.710,   6.710,   6.774,   6.774,
     .   6.831,   6.831,   6.919,   6.919,   6.971,   6.971,   6.963,   6.963/
      data (latitude(i),i= 8961, 9040)/
     .   7.028,   7.028,   7.093,   7.093,   3.292,   3.292,  11.694,  11.694,
     .  11.595,  11.595,  11.212,  11.212,  11.502,  11.502,  11.246,  11.246,
     .  11.119,  11.119,  10.779,  10.779,  10.567,  10.567,  10.830,  10.830,
     .  10.145,  10.145,  10.085,  10.085,   8.303,   8.303,  10.057,  10.057,
     .  10.137,  10.137,  10.949,  10.949,  10.864,  10.864,  -7.436,  -7.436,
     .  -6.653,  -6.653,  -6.389,  -6.389,  -6.229,  -6.229,  -6.196,  -6.196,
     .  -6.123,  -6.123,  -6.168,  -6.168,  -5.328,  -5.328,  -5.360,  -5.360,
     .  -5.235,  -5.235,  -5.251,  -5.251,  -5.237,  -5.237,  -5.285,  -5.285,
     .  -5.440,  -5.440,  -5.408,  -5.408,   7.600,   7.098,   6.604,   6.126,
     .   5.936,   6.102,   6.324,   6.717,   7.213,   7.677,   8.104,   8.252/
      data (latitude(i),i= 9041, 9120)/
     .   8.251,   8.705,   9.111,   9.499,   9.821,   9.482,   9.594,   9.094,
     .   8.614,   8.122,   7.622,   7.601,   7.683,   7.679,   9.102,   9.102,
     .   9.535,   9.535,   9.760,   9.760,  21.885,  21.884,  12.895,  12.897,
     .  12.307,  12.307,  12.179,  12.069,  21.995,  21.995,  22.044,  22.044,
     .  22.037,  22.037,  21.963,  21.963,  22.158,  22.158,  22.775,  22.302,
     .  22.727,  22.775,  22.446,  22.446,  22.877,  22.877,  23.034,  23.034,
     .  23.058   23.058,  22.606,  22.606,  22.449,  22.449,  22.516,  22.516,
     .  22.355,  22.355,  22.251,  22.251,  22.197,  22.197,  21.913,  21.911,
     .  21.755,  21.755,  23.506,  23.506,  20.084,  20.081,  19.951,  19.951,
     .  19.560,  19.560,  18.890,  18.885,  14.139,  14.134,  14.902,  14.900/
      data (latitude(i),i= 9121, 9200)/
     .  16.231,  16.228,  15.933,  15.924,   7.605,   7.605,   9.411,   9.411,
     .   9.471,   9.471,  16.507,  16.506,  15.565,  15.561,  14.179,  14.180,
     .  13.939,  13.934,  13.859,  13.859,  13.444,  13.444,  12.974,  12.975,
     .  12.773,  12.774,  12.361,  12.368,  12.502,  12.506,  12.417,  12.423,
     .  12.371,  12.375,  12.289,  12.294,  12.169,  12.174,  12.544,  12.548,
     .  11.844,  11.846,  11.780,  11.782,  11.655,  11.661,  11.893,  11.904,
     .  11.608,  11.611,  11.928,  11.930,  11.858,  11.864,  11.453,  11.461,
     .  11.224,  11.229,  11.006,  11.006,  10.831,  10.833,  10.976,  10.978,
     .  10.880,  10.881,  10.441,  10.436,  10.521,  10.520,  10.163,  10.163,
     .   9.803,   9.797,  10.007,  10.007,  10.727,  10.725,  13.664,  13.662/
      data (latitude(i).i= 9201, 9280)/
     .  13.532,  13.034,  12.532,  12.081,  11.579,  12.069,  12.568,  13.063,
     .  13.544,  13.533,  12.960,  12.951,  12.144,  12.145,  12.015,  12.015,
     .  12.271,  12.269,  11.580,  11.576,  11.468,  11.468,  10.786,  10.777,
     .   9.239,   9.239,   8.323,   8.324,   8.545,   8.542,   8.214,   8.209,
     .   8.007,   7.999,   8.014,   8.012,   7.418,   7.413,   7.414,   7.410,
     .   7.190,   7.182,  13.174,  13.175,   9.762,   9.756,   9.081,   9.072,
     .  12.948,  12.948,   8.170,   8.170,   8.108,   8.100,   8.166,   8.162,
     .   7.266,   7.261,   6.571,   6.571,   7.174,   7.168,   9.550,   9.544,
     .   9.793,   9.788,   9.542,   9.542,   9.623,   9.620,   6.709,   6.703,
     .  12.298,  12.297,  12.140,  12.142,  11.753,  11.754,   6.260,   6.259/
      data (latitude(i),i= 9281, 9360)/
     .   6.464,   6.464,   5.449,   5.446,   2.871,   2.868,   1.388,   1.382,
     .  11.406,  11.403,  10.442,  10.439,   8.766,   8.765,   9.926,   9.926,
     .  10.659,  10.662,  10.553,  10.552,   7.872,   7.872,   9.708,   9.708,
     .   9.419,   9.419,  20.871,  20.873,  20.834,  20.836,  20.844,  20.840,
     .  21.034,  21.032,  20.958,  20.956,  20.136,  20.139,  21.036,  21.037,
     .  21.016,  21.017,  21.113,  21.118,  21.273,  21.271,  21.354,  21.358,
     .  21.384,  21.386,  15.790,  15.790,  16.051,  16.051,  16.666,  16.666,
     .  16.839,  16.839,  16.936,  16.936,  16.983,  16.983,  16.982,  16.982,
     .  16.463,  16.463,  16.544,  16.544,  16.447,  16.447,  20.720,  20.720,
     .  21.054,  21.055,  19.977,  19.966,  20.106,  19.749,  19.351,  18.883/
      data (latitude(i),i= 9361, 9440)/
     .  18.533,  18.199,  18.288,  18.496,  18.994,  19.451,  19.717,  19.983,
     .  21.056,  21.058,  21.196,  21.199,  24.434,  24.439,  25.425,  25.426,
     .  29.079,  29.086,  29.681,  29.686,  30.063,  30.067,  31.799,  31.580,
     .  31.782,  31.801,  37.978,  37.979,  37.811,  37.804,  34.499,  34.492,
     .  33.542,  33.226,  33.544,  33.536,  34.915,  34.902,  34.950,  34.943,
     .  37.528,  37.518,  25.119,  25.120,  24.618,  24.141,  23.651,  23.162,
     .  22.728,  22.286,  22.487,  22.954,  23.453,  23.931,  24.373,  24.778,
     .  25.122,  22.045,  22.045,  23.534,  23.536,  24.420,  24.419,  32.454,
     .  32.867,  33.246,  33.609,  33.933,  33.574,  33.354,  32.883,  32.837,
     .  32.694,  34.085,  33.620,  33.501,  33.285,  32.845,  33.106,  33.605/
      data (latitude(i),i= 9441, 9520)/
     .  34.016,  33.977,  34.326,  34.200,  34.065,  41.439,  40.976,  40.499,
     .  40.038,  39.556,  39.089,  38.668,  38.280,  37.784,  37.287,  36.821,
     .  36.339,  35.851,  35.436,  35.037,  35.538,  35.183,  35.007,  34.953,
     .  34.646,  34.646,  34.854,  34.355,  34.170,  33.724,  33.538,  33.942,
     .  34.415,  34.716,  34.728,  34.513,  34.276,  34.241,  33.842,  34.034,
     .  34.004,  34.406,  34.704,  35.040,  35.438,  35.563,  35.493,  35.628,
     .  35.724,  35.535,  35.973,  36.356,  36.770,  37.273,  37.521,  37.141,
     .  36.750,  37.024,  37.249,  37.637,  37.991,  38.449,  38.888,  39.366,
     .  39.868,  40.371,  40.824,  40.983,  41.485,  41.434,  45.513,  45.125,
     .  44.710,  44.378,  44.095,  43.923,  44.188,  43.687,  43.252,  42.979/
      data (latitude(i),i= 9521, 9600)/
     .  42.969,  42.663,  42.221,  42.194,  42.461,  42.559,  42.565,  42.068,
     .  41.649,  42.069,  42.542,  42.869,  43.367,  43.158,  43.621,  44.076,
     .  44.574,  45.067,  45.506,  43.587,  43.587,  34.688,  34.686,  34.237,
     .  34.230,  33.830,  33.824,  33.311,  33.307,  33.077,  33.069,  34.538,
     .  34.530,  34.779,  34.777,  33.144,  33.141,  34.096,  34.086,  45.243,
     .  45.246,  45.451,  45.449,  42.211,  42.214,  38.287,  37.811,  38.300,
     .  38.281,  36.328,  36.318,  32.694,  32.219,  31.749,  31.249,  31.642,
     .  31.161,  31.384,  31.868,  32.351,  32.458,  32.349,  32.356,  32.693,
     .  32.697,  30.828,  30.355,  30.838,  30.823,  30.445,  30.443,  29.993,
     .  29.992,  29.874,  29.872,  29.653,  29.653,  28.356,  28.356,  28.500/
      data (latitude(i),i= 9601, 9680)/
     .  28.223,  28.509,  28.498,  28.191,  28.191,  27.899,  27.901,  27.415,
     .  27.417,  26.859,  26.456,  26.876,  26.855,  26.386,  26.386,  25.869,
     .  25.870,  24.866,  24.859,  24.438,  24.439,  24.411,  24.405,  24.469,
     .  24.470,  24.483,  24.483,  30.480,  30.480,  27.733,  27.733,  27.621,
     .  27.621,  27.174,  27.174,  27.079,  27.079,  26.712,  26.712,  24.789,
     .  24.789,  25.446,  25.446,  20.024,  20.024,  27.236,  27.236,  44.509,
     .  44.052,  44.507,  43.536,  43.535,  43.866,  43.861,  53.153,  52.650,
     .  52.154,  51.658,  51.215,  50.719,  50.217,  49.716,  49.214,  48.736,
     .  48.272,  47.774,  47.276,  46.776,  46.274,  46.608,  46.588,  46.119,
     .  46.616,  47.008,  47.459,  47.944,  48.432,  48.910,  49.304,  49.156/
      data (latitude(i),i= 9681, 9760)/
     .  48.753,  49.204,  49.690,  50.169,  50.656,  51.152,  51.641,  52.117,
     .  52.604,  52.803,  48.291,  48.291,  46.211,  46.243,  52.803,  53.301,
     .  53.768,  54.268,  53.782,  53.461,  53.153,  54.903,  54.907,  54.774,
     .  54.939,  54.782,  54.776,  59.018,  59.023,  59.317,  59.318,  45.514,
     .  45.199,  44.961,  44.550,  45.012,  45.416,  45.503,  45.514,  45.603,
     .  45.866,  45.603,  47.127,  47.125,  47.306,  47.303,  47.721,  47.723,
     .  48.049,  48.051,  48.735,  48.736,  49.168,  49.169,  49.291,  49.296,
     .  49.767,  49.767,  50.879,  50.884,  50.054,  50.299,  50.094,  50.052,
     .  50.629,  50.627,  55.327,  54.829,  55.326,  54.863,  54.510,  54.865,
     .  54.865,  58.559,  58.809,  58.562,  58.561,  70.799,  70.997,  71.002/
      data (latitude(i),i= 9761, 9840)/
     .  71.501,  71.556,  71.536,  71.191,  70.799,  69.768,  69.978,  69.567,
     .  69.763,  70.658,  70.659,  70.737,  70.746,  70.821,  70.819,  71.509,
     .  71.506,  74.248,  74.246,  73.892,  73.901,  73.875,  73.455,  73.290,
     .  73.436,  73.848,  73.878,  75.586,  75.090,  74.786,  74.855,  75.238,
     .  75.336,  75.589,  76.634,  76.637,  74.807,  74.965,  74.922,  75.297,
     .  75.027,  75.462,  75.861,  76.082,  75.622,  75.997,  75.967,  75.468,
     .  74.981,  74.660,  74.825,  75.374,  75.379,  72.429,  72.235,  72.503,
     .  72.429,  72.629,  72.632,  72.651,  72.667,  72.659,  72.653,  72.871,
     .  72.871,  72.401,  72.743,  73.063,  73.431,  73.056,  72.566,  72.402,
     .  72.971,  72.779,  72.471,  72.887,  73.385,  73.676,  73.558,  73.055/
      data (latitude(i),i= 9841, 9920)/
     .  72.973,  73.838,  73.838,  73.039,  73.041,  74.296,  74.296,  74.093,
     .  74.094,  76.368,  76.370,  76.605,  76.609,  77.231,  77.231,  77.387,
     .  77.390,  78.127,  78.127,  78.214,  78.214,  77.954,  78.190,  78.303,
     .  78.751,  79.149,  79.075,  78.589,  78.116,  77.955,  79.249,  79.743,
     .  80.098,  80.101,  79.651,  79.174,  78.929,  78.832,  79.247,  81.290,
     .  80.978,  80.495,  80.029,  80.213,  80.662,  81.132,  81.290,  81.206,
     .  81.209,  80.954,  80.955,  79.544,  79.544,  80.044,  79.834,  80.062,
     .  80.044,  77.164,  77.199,  77.202,  77.199,  76.717,  76.717,  76.263,
     .  76.263,  76.706,  76.706,  75.993,  75.993,  75.928,  75.928,  75.500,
     .  75.496,  75.138,  75.138,  74.989,  74.986,  74.810,  74.813,  74.506/
      data (latitude(i),i= 9921,10000)/
     .  74.506,  74.543,  74.541,  74.500,  74.503,  74.025,  74.026,  74.079,
     .  74.084,  74.159,  74.159,  74.606,  74.605,  72.893,  72.902,  72.329,
     .  72.344,  73.172,  73.182,  73.527,  73.528,  73.410,  73.420,  73.026,
     .  73.033,  73.038,  73.492,  73.015,  73.051,  69.892,  70.394,  70.095,
     .  69.902,  69.181,  69.196,  69.083,  69.509,  69.241,  68.815,  69.050,
     .  69.089,  70.977,  70.987,  71.781,  72.153,  72.645,  73.136,  73.386,
     .  73.238,  72.802,  72.301,  71.924,  71.925,  71.462,  71.023,  70.573,
     .  70.683,  70.752,  71.203,  71.576,  71.805,  74.687,  74.432,  73.960,
     .  73.474,  73.304,  73.584,  74.086,  74.482,  74.574,  74.574,  74.720,
     .  75.161,  75.483,  75.719,  75.947,  76.234,  76.670,  76.951,  76.532/
      data (latitude(i),i=10001,10080)/
     .  76.530,  76.310,  76.260,  76.060,  75.726,  75.385   74.999,  74.689,
     .  80.007,  80.007,  80.107,  80.111,  80.122,  80.127,  80.209,  80.219,
     .  80.089,  80.520,  80.075,  80.098,  80.629,  80.808,  80.592,  80.627,
     .  80.541,  80.887,  80.454,  80.541,  80.839,  81.166,  80.718,  80.844,
     .  81.004,  81.007,  81.103,  81.103,  80.827,  80.827,  81.023,  80.851,
     .  81.024,  80.719,  80.723,  81.216,  81.219,  81.473,  81.473,  81.730,
     .  81.730,  81.608,  81.608,  81.686,  81.691,  66.791,  66.790,  65.189,
     .  65.181,  59.043,  59.042,  59.084,  59.082,  58.675,  58.670,  58.150,
     .  58.143,  58.597,  58.220,  58.590,  46.244,  46.211,  46.366,  46.146,
     .  46.348,  46.366,  45.073,  45.073,  39.593,  39.593,  39.092,  39.092/
      data (latitude(i),i=10081,10160)/
     .  45.044,  45.044,  45.579,  45.579,  39.328,  38.942,  38.440,  37.939,
     .  37.532,  37.476,  37.111,  36.830,  36.660,  36.601,  36.728,  36.861,
     .  36.913,  37.415,  37.915,  38.419,  38.909,  39.005,  33.316,  33.311,
     .  13.262,  12.956,  12.579,  12.791,  13.192,  13.262,  51.508,  51.035,
     .  50.554,  51.045,  51.515,  55.724,  55.255,  54.760,  54.268,  53.799,
     .  53.326,  52.981,  52.705,  52.440,  52.111,  51.705,  51.546,  51.511,
     .  51.868,  52.089,  52.505,  52.817,  53.247,  53.632,  53.996,  54.462,
     .  54.912,  55.391,  55.746,  46.429,  46.774,  46.727,  46.798,  46.594,
     .  46.633,  46.473,  46.792,  46.378,  46.379,  46.458,  46.575,  46.465,
     .  46.328,  46.016,  45.533,  45.028,  45.438,  45.942,  46.368,  46.429/
      data (latitude(i),i=10161,10240)/
     .  44.736,  45.158,  45.481,  45.953,  46.397,  46.765,  46.679,  62.861,
     .  62.586,  62.089,  61.631,  61.144,  61.049,  61.530,  61.886,  62.384,
     .  62.094,  62.572,  62.877,  62.862,  58.952,  58.555,  58.091,  58.441,
     .  58.867,  58.571,  59.062,  58.938,  61.731,  61.394,  61.092,  60.627,
     .  60.171,  60.058,  60.432,  60.894,  61.384,  61.729,  60.592,  60.592,
     .  69.287,  69.286,  46.680,  46.195,  46.080,  46.220,  46.165,  45.811,
     .  45.379,  45.725,  46.036,  46.220,  39.328,  39.827,  40.264,  40.222,
     .  40.586,  40.866,  41.321,  41.738,  42.116,  42.534,  42.958,  43.456,
     .  43.952,  44.366,  44.747,  44.747,  45.163,  45.633,  45.767,  46.057,
     .  46.469,  46.614,  46.882,  47.113,  46.884,  46.949,  46.658,  46.155/
      data (latitude(i),i=10241,10320)/
     .  46.031,  46.033,  45.592,  45.380,  45.318,  44.906,  44.571,  44.634,
     .  44.130,  43.661,  43.169,  42.901,  42.812,  42.312,  41.814,  41.367,
     .  41.863,  42.109,  41.794,  41.343,  40.982,  40.680,  40.886,  40.998,
     .  40.524,  40.023,  40.004,  39.545,  39.253,  39.056,  39.005,  46.079,
     .  45.938,  45.477,  45.008,  44.508,  44.020,  43.745,  43.663,  43.603,
     .  43.754,  44.257,  44.678,  44.736, -21.008, -21.503, -21.966, -22.427,
     . -22.411, -22.634, -23.138, -23.572, -23.949, -24.151, -24.151, -24.605,
     . -24.978, -25.312, -25.807, -26.305, -26.802, -27.295, -27.723, -27.953,
     . -27.952, -28.444, -28.943, -29.425, -29.920, -30.396, -30.895, -31.388,
     . -31.863, -32.336, -32.444, -32.443, -32.796, -33.122, -33.554, -34.009/
      data (latitude(i),i=10321,10400)/
     . -34.490, -34.980, -35.406, -35.860, -36.087, -36.086, -36.591, -37.078,
     . -37.565, -37.793, -37.814, -37.914, -38.209, -38.578, -38.695, -39.143,
     . -39.143, -38.819, -38.464, -38.409, -37.933, -38.109, -38.517, -38.858,
     . -38.861, -38.859, -38.629, -38.375, -38.268, -38.144, -38.008, -37.611,
     . -37.220, -36.956, -36.957, -36.460, -36.013, -35.659, -36.001, -35.633,
     . -35.653, -35.653, -35.649, -35.167, -34.665, -34.216, -34.669, -35.152,
     . -35.243, -34.825, -34.324, -33.838, -33.376, -32.876, -33.182, -33.648
     . -33.878, -34.243, -34.630, -35.013, -35.010, -34.733, -34.238, -33.791,
     . -33.337, -33.106, -32.606, -32.270, -32.211, -31.954, -32.029, -32.028,
     . -31.833, -31.574, -31.605, -31.576, -31.626, -31.680, -31.899, -32.094/
      data (latitude(i),i=10401,10480)/
     . -32.239, -32.307, -32.247, -32.274, -40.745, -41.206, -41.712, -42.215,
     . -42.604, -43.105, -43.231  -43.233, -42.825, -43.262, -43.614, -43.539,
     . -43.104, -42.673, -42.170, -41.745, -41.405, -41.405, -40.909, -40.711,
     . -41.011, -41.171, -40.979, -40.831, -40.744, -32.272, -32.551, -32.764,
     . -32.945, -33.331, -33.774, -34.003, -33.894, -33.826, -33.819, -33.885,
     . -33.937, -34.097, -34.389, -34.388, -34.452, -34.739, -35.015, -35.021,
     . -35.035, -34.850, -34.517, -34.362, -33.879, -33.522, -33.524, -33.417,
     . -32.919, -32.424, -31.918, -31.441, -30.987, -30.533, -30.040, -29.540,
     . -29.064, -28.789, -28.789, -28.321, -27.878, -27.378, -26.932, -26.533,
     . -26.048, -26.485, -25.984, -26.313, -25.888, -25.472, -25.037, -24.894/
      data (latitude(i),i=10481,10560)/
     . -24.894, -24.435, -23.935, -23.510, -23.009, -22.520, -22.050  -22.509,
     . -22.149, -21.753, -21.569, -21.260, -20.999, -20.999, -20.747, -20.624,
     . -20.661, -20.410, -20.311, -20.022, -20.069, -19.973, -19.973, -19.907,
     . -19.715, -19.465, -19.061, -18.625, -18.270, -17.767, -17.266, -16.937,
     . -16.521, -16.596, -16.596, -17.018, -17.464, -17.058, -16.651, -16.157,
     . -16.273, -16.399, -15.896, -15.399, -15.521, -15.520, -15.519, -15.156,
     . -15.141, -14.639, -14.232, -14.641, -14.196, -14.107, -13.727, -13.726,
     . -13.950, -14.318, -14.676, -15.179, -14.794, -14.848, -14.927, -14.790,
     . -14.526, -14.027, -13.579, -13.324, -12.923, -12.923, -12.711, -12.251,
     . -12.296, -12.291, -12.138, -11.671, -11.489, -11.512, -11.499, -11.499/
      data (latitude(i),i=10561,10640)/
     . -11.792, -11.905, -12.062, -12.147, -12.111, -11.819, -12.299, -12.459,
     . -11.989, -11.882, -11.882, -12.269, -12.714, -13.183, -13.278, -13.780,
     . -14.236, -14.660, -14.724  -14.726, -15.103, -15.372, -15.724, -15.896,
     . -16.159, -16.454, -16.751, -16.959, -17.225, -17.512, -17.572, -17.571,
     . -17.688, -17.470, -17.001, -16.573, -16.103, -15.602, -15.131, -14.63
     . -14.134, -13.640, -13.352, -13.352, -12.859, -12.358, -11.931, -11.452,
     . -10.953, -10.744, -11.216, -11.710, -11.973, -11.974  -12.474, -12.925,
     . -13.419, -13.920, -14.395, -14.300, -14.561, -14.842, -14.946, -14.944,
     . -15.434, -15.922, -16.421, -16.816, -17.259, -17.745, -18.238, -18.532,
     . -18.532, -19.026, -19.298, -19.415, -19.829, -19.947, -20.198, -20.699/
      data (latitude(i),i=10641,10720)/
     . -21.000, -21.007, -12.183, -12.183, -12.163, -12.163,   4.170,   4.169,
     .   4.380,   4.741,   4.939,   5.110,   5.494,   5.782,   5.749,   5.867,
     .   5.867,   6.295,   6.625,   6.586,   6.399,   5.981,   5.550,   5.354,
     .   4.902,   4.902,   4.818,   4.891,   4.590,   4.149,   3.790,   3.394,
     .   3.085,   2.963,   2.869,   2.547,   2.071,   1.632,   1.626,   1.695,
     .   2.038,   2.064,   4.493,   4.492,   4.675,   4.676,   6.671,   6.672,
     .   7.294,   7.298,   7.232,   7.241,   5.276,   5.276,   2.490,   2.497,
     .   4.903,   4.818,   4.891,   4.711,   4.590,   5.571,   5.582,   5.274,
     .   5.218,   5.221,   5.103,   4.725,   4.322,   3.937,   3.653,   3.384,
     .   3.074,   2.602,   2.301,   2.026,   1.821,   1.821,   1.702,   1.534/
      data (latitude(i),i=10721,10800)/
     .   1.119,   0.741,   0.609,   0.245,   0.398,   0.341,  -0.141,  -0.257,
     .  -0.259,  -0.743,  -1.033,  -1.038,  -1.519,  -1.995,  -2.483,  -2.349,
     .  -2.623,  -2.966,  -3.453,  -3.953,  -4.458,  -4.958,  -5.457,  -5.660,
     .  -5.660,  -5.743,  -5.487,  -5.453,  -5.051,  -4.796,  -4.597,  -4.182,
     .  -3.793,  -3.442,  -3.100,  -2.680,  -2.315,  -1.825,  -1.368,  -1.044,
     .  -1.043,  -0.631,  -0.240,   0.149,   0.496,   0.968,   1.438,   1.894,
     .   2.163,   2.369,   2.867,   3.231,   3.606,   3.800,   4.162,   4.512,
     .   4.889,   5.360,   5.569,   5.706,   5.703,   5.839,   5.838,   2.359,
     .   2.596,   2.385,   2.355,   2.054,   2.054,   2.057,   2.054,   1.625,
     .   1.623,  -0.317,  -0.321,  -0.460,  -0.466,  -1.773,  -1.494,  -1.008
      data (latitude(i),i=10801,10880)/
     .  -1.286,  -1.737,  -1.772,  -2.374,  -2.376,  -2.757,  -2.759,  -3.067,
     . -3.067,  -5.485,  -5.487,   0.933,   0.950,   1.369,   1.265,   0.926,
     . -5.744,  -5.744,  -6.534,  -6.533,  -2.825,  -2.825,  -2.903,  -2.901,
     . -1.491,  -1.904,  -2.393,  -2.560,  -3.059,  -2.903,  -2.517,  -2.122,
     . -2.072,  -1.634,  -1.500,  -1.492,  -2.558,  -2.681,  -3.169,  -3.047,
     . -2.558,  -2.557,  -0.270,  -0.273,  -0.388,  -0.389,  -0.347,  -0.348,
     . -0.002,  -0.005,   0.212,   0.209,   0.162,   0.159,   0.366,   0.360,
     .  0.708,   0.706,   0.835,   0.835,   0.974,   0.969,   1.025,   1.024,
     .  1.142,   1.135,   0.956,   0.946,   0.791,   0.789,   0.794,   0.786,
     .  0.797,   0.795,   0.742,   0.740,   1.025,   1.025,   0.593,   0.589/
      data (latitude(i),i=10881,10960)/
     .  0.859,   1.023,   0.877,   0.854,   0.796,   0.773,   0.799,   0.786,
     .  1.2[illegible],   1.206,   1.448,   1.573,   1.451,   1.436,   1.945,   1.940,
     .  2.838,   2.844,   2.736,   2.743,   3.281,   3.286,   3.154,   3.155,
     .  4.692,   4.693,   3.850,   3.852,   2.982,   2.984,   2.898,   2.900,
     .  2.533,   2.540,   1.013,   1.014,  -1.031,  -1.033,  -1.570,  -1.571,
     . -4.760,  -4.764,  -3.281,  -3.780,  -3.300,  -3.285,  -3.405,  -3.416,
     .  2.313,   2.313,   3.430,   3.433,   3.578,   3.580,   3.572,   3.574,
     .  4.102,   4.102,   4.258,   4.261,   2.064,   1.718,   1.292,   0.822,
     .  0.323,  -0.105,  -0.588,  -0.969,  -1.293,  -1.782,  -2.248,  -2.521,
     . -2.521,  -2.901,  -3.051,  -3.009,  -3.446,  -3.319,  -3.281,  -3.254/
      data (latitude(i),i=10961,11040)/
     . -3.469,  -3.334,  -3.349,  -3.349,  -3.852,  -3.979,  -3.794,  -3.546,
     . -3.124,  -2.634,  -2.191,  -1.812,  -1.427,  -1.165,  -0.880,  -0.525,
     . -0.524,  -0.028,   0.470,   0.784,   0.829,   0.956,   1.311,   1.598,
     .  1.956,   2.444,   2.804,   3.195,   3.600,   3.704,   4.169,  -7.684,
     . -7.684,  -7.838,  -8.059,  -8.206,  -8.264,  -8.270,  -8.343,  -8.398,
     . -8.356,  -8.554,  -8.639,  -8.140,  -7.708,  -7.719,  -7.730,  -7.347,
     . -7.046,  -7.046,  -6.91[illegible],  -6.724,  -6.700,  -6.501,  -6.952,  -6.927,
     . -6.848,  -6.850,  -6.550,  -6.341,  -6.206,  -5.981,  -6.019,  -6.018,
     . -6.042,  -5.903,  -6.367,  -6.763,  -6.829,  -6.976,  -7.422,  -7.481,
     . -7.656,  -7.787,  -7.676,  -7.694,  -7.796,  -7.840,  -5.732,  -5.731/
      data (latitude(i),i=11041,11120)/
     . -6.839,  -6.839,  -8.092,  -8.187,  -8.173,  -8.640,  -8.432,  -8.117,
     . -8.090,  -8.669,  -8.669,  -8.219,  -8.638,  -8.919,  -8.434,  -8.218,
     . -8.386,  -8.573,  -8.659,  -8.330,  -8.372,  -8.297,  -8.626,  -8.748,
     . -8.841,  -8.923,  -9.035,  -8.741,  -8.375,  -8.387,  -8.134,  -8.136,
     . -9.433,  -9.374,  -9.412,  -9.662, -10.032, -10.260,  -9.974,  -9.755,
     . -9.623,  -9.429,  -8.430,  -8.430,  -8.268,  -8.328,  -8.467,  -8.489,
     . -8.641,  -8.385,  -8.749,  -8.849,  -8.914,  -8.827,  -8.821,  -8.784,
     . -8.366,  -8.269,  -8.289,  -8.286,  -8.273,  -8.226,  -8.567,  -8.271,
     . -8.345,  -8.342,  -8.136,  -8.167,  -8.446,  -8.136, -10.547, -10.545,
     .-10.761, -10.509, -10.886, -10.760, -10.257, -10.257, -10.156, -10.153/
      data (latitude(i),i=11121,11200)/
     . -9.462,  -9.226,  -9.074,  -8.911,  -8.686,  -8.459,  -8.510,  -8.561,
     . -8.820,  -9.125,  -9.182,  -9.184,  -9.354,  -9.745, -10.226, -10.279,
     .-10.107,  -9.741,  -9.461,  -8.144,  -8.150,  -7.661,  -7.606,  -7.888,
     . -7.658,  -7.512,  -7.512,  -8.105,  -8.104,  -7.065,  -7.065,  -7.798,
     . -7.799,  -7.116,  -7.621,  -8.020,  -7.526,  -7.142,  -7.117,  -8.084,
     . -8.085,  -7.116,  -7.115,  -5.784,  -6.286,  -5.786,  -5.787,  -5.591,
     . -5.653,  -5.193,  -4.711,  -4.219,  -3.736,  -3.517,  -3.083,  -2.619,
     . -2.160,  -1.676,  -1.178,  -0.771,  -0.878,  -0.877,  -0.379,   0.110,
     .  0.583,   0.808,   1.182,   1.247,   1.081,   1.009,   0.871,   0.815,
     .  0.816,   0.913,   0.829,   1.017,   1.412,   1.73[illegible],   1.238,   0.858/
      data (latitude(i),i=11201,11280)/
     .  0.452,   0.333,   0.304,   0.511,   0.511,   0.493,   0.480,   0.529,
     .  0.410,   0.528,   0.240,  -0.232,  -0.719,  -1.018,  -1.368,  -1.367,
     . -1.191,  -0.849,  -0.931,  -0.771,  -0.601,  -0.947,  -0.905,  -1.283,
     . -1.616   -1.858,  -1.994,  -1.994,  -2.254,  -2.695,  -3.081,  -3.565,
     . -3.884,  -4.284,  -4.429,  -4.827,  -4.764,  -4.268,  -4.044,  -4.045,
     . -3.796,  -3.349,  -2.867,  -2.678,  -2.965,  -3.422,  -3.919,  -4.421,
     . -4.918,  -5.412,  -5.592,   1.521,   1.526,   2.333,   2.329,   2.759,
     .  2.753,   3.423,   3.423,   3.757,   3.754,   3.903,   3.900,   4.477,
     .  4.006,   4.508,   4.471,  -0.146,  -0.152,  -0.472,  -0.474,  -0.377,
     . -0.379,  -0.336,  -0.345,  -0.266,  -0.265,  -1.305,  -1.624,  -1.212/
```

```
      data (latitude(i),i=11281,11360)/
     .  -1.226,  -1.311,  -1.841,  -1.843,  -1.689,  -1.691,  -1.949,  -1.954,
     .  -1.658,  -1.700,  -2.013,  -1.657,  -1.784,  -1.791,  -1.953,  -1.784,
     .  -1.979,  -2.467,  -2.004,  -1.979,  -3.549,  -3.547,  -3.978,  -3.979,
     .  -5.078,  -5.081,  -4.618,  -5.115,  -4.616,  -4.620,  -4.404,  -4.906,
     .  -5.402,  -5.684,  -5.198,  -4.697,  -4.406,  -3.120,  -3.061,  -3.219,
     .  -3.711,  -3.767,  -3.468,  -3.123,  -2.860,  -2.855,  -2.959,  -2.834,
     .  -2.989,  -3.195,  -3.621,  -3.629,  -3.364,  -3.445,  -3.351,  -3.456,
     .  -3.446,  -2.959,  -2.859,  -3.594,  -3.593,  -3.523,  -3.520,  -3.496,
     .  -3.494,  -3.648,  -3.648,  -4.503,  -4.510,  -3.963,  -3.965,  -4.432,
     .  -4.437,  -4.694,  -4.694,  -5.677,  -5.679,  -5.301,  -5.755,  -5.293/
      data (latitude(i),i=11361,11440)/
     .  -5.304,  -6.170,  -6.512,  -6.922,  -6.427,  -6.172,  -5.434,  -5.872,
     .  -6.356,  -5.956,  -5.484,  -5.437,  -1.435,  -1.480,  -1.739,  -1.435,
     .  -1.176,  -1.178,  -0.309,  -0.711,  -0.566,  -0.310,  -0.615,  -0.614,
     .  -0.266,  -0.268,   0.378,   0.374,   0.740,   0.738,   0.852,   0.851,
     .   2.014,   2.506,   2.010,   2.017,   0.202,   0.398,   0.468,  -0.032,
     .  -0.498,  -0.882,  -0.611,  -0.226,  -0.214,  -0.214,   0.288,   0.774,
     .   1.240,   1.718,   2.118,   1.634,   1.433,   1.432,   1.042,   1.159,
     .   1.532,   1.066,   0.824,   0.551,   0.201,   0.033,   0.032,  -0.010,
     .  -0.164,  -0.244,  -0.240,  -0.053,  -0.009,  -0.418,  -0.419,  -0.770,
     .  -0.773,  -0.893,  -0.891,  -1.146,  -1.146,  -1.680,  -2.008,  -1.678/
      data (latitude(i),i=11441,11520)/
     .  -1.685,  -2.625,  -2.621,  -4.105,  -4.107,  -7.379,  -7.559,  -8.041,
     .  -8.348,  -8.380,  -7.894,  -7.503,  -7.377,  -8.168,  -8.177,  -9.128,
     .  -8.810,  -8.438,  -7.989,  -8.154,  -8.165,  -7.662,  -7.255,  -6.976,
     .  -6.974,  -6.974,  -6.743,  -6.341,  -5.841,  -5.463,  -5.213,  -4.996,
     .  -4.845,  -4.669,  -4.495,  -4.468,  -4.259,  -4.259,  -3.924,  -3.827,
     .  -3.394,  -3.799,  -4.106,  -3.685,  -3.194,  -2.922,  -2.801,  -2.471,
     .  -2.565,  -2.718,  -2.718,  -2.292,  -2.240,  -2.283,  -2.222,  -1.988,
     .  -1.527,  -1.410,  -0.910,  -0.789,  -0.789,  -0.584,  -0.351,  -0.453,
     .  -0.697,  -0.753,  -1.161,  -1.662,  -2.161,  -2.614,  -2.861,  -2.863,
     .  -3.168,  -3.386,  -3.131,  -2.708,  -2.328,  -2.204,  -1.954,  -1.954/
      data (latitude(i),i=11521,11600)/
     .  -1.591,  -1.596,  -1.761,  -1.974,  -2.181,  -2.375,  -2.408,  -2.613,
     .  -2.608,  -1.748,  -1.751,  -2.013,  -2.018,  -2.324,  -2.334,  -0.941,
     .  -0.943,  -1.480,  -1.481,  -1.597,  -1.647,  -1.722,  -1.818,  -1.688,
     .  -1.600,  -0.658,  -0.784,  -1.093,  -1.142,  -0.771,  -0.662,  -2.608,
     .  -2.761,  -2.964,  -3.161,  -3.317,  -3.403,  -3.641,  -3.809,  -4.041,
     .  -4.373,  -4.619,  -5.057,  -5.506,  -5.651,  -5.898,  -5.971,  -6.330,
     .  -6.751,  -6.740,  -6.748,  -6.747,  -7.217,  -7.602,  -7.946,  -8.290,
     .  -8.721,  -9.098,  -9.026,  -9.525,  -9.620, -10.078, -10.209, -10.689,
     . -10.544, -10.463, -10.463, -10.345, -10.238, -10.181, -10.138,  -9.965,
     .  -9.546,  -9.246,  -8.883,  -8.455,  -8.366,  -8.364,  -8.040,  -7.914/
      data (latitude(i),i=11601,11680)/
     .  -7.777,  -7.576,  -7.704,  -8.005,  -8.299,  -8.283,  -8.464,  -8.869,
     .  -9.145,  -9.252,  -9.192,  -9.177,  -9.129,  -8.426,  -8.696,  -8.451,
     .  -8.426,  -8.359,  -8.358, -10.539, -10.539, -11.323, -11.519, -11.363,
     . -11.323, -11.353, -11.353, -10.623, -10.623, -10.598, -10.598,  -9.712,
     .  -9.929,  -9.759,  -9.712,  -9.342,  -9.627,  -9.456,  -9.342,  -8.977,
     .  -8.977,  -8.432,  -8.432,  -9.214,  -9.214,  -5.194,  -5.194,  -5.492,
     .  -5.492,  -4.146,  -4.602,  -4.987,  -5.486,  -5.540,  -5.938,  -6.134,
     .  -6.296,  -6.290,  -6.058,  -5.918,  -5.749,  -5.533,  -5.586,  -5.527,
     .  -5.064,  -5.546,  -5.551,  -5.227,  -4.935,  -4.456,  -4.206,  -4.146,
     .  -2.564   -2.873,  -3.062,  -3.328,  -3.651,  -3.987,  -4.450,  -4.141
      data (latitude(i),i=11681,11760)
     .  -3.728,  -3.456,  -3.147,  -2.874,  -2.564,  -2.909,  -2.909,  -2.466,
     .  -2.518,  -2.478,  -2.460,  -1.353,  -1.353,  -2.596,  -2.596,  -2.712,
     .  -2.712,  -3.049,  -3.049,  -3.446,  -3.446,  -4.040,  -4.040,  -4.371,
     .  -4.371,  -3.435,  -3.435,  -4.540,  -4.540,  -5.436,  -5.683,  -6.096,
     .  -6.395,  -6.882,  -6.644,  -6.219,  -5.801,  -5.436,  -5.019,  -5.019,
     .  -4.648,  -4.648,  -4.868,  -4.868,  -2.237,  -2.237,  -1.961,  -2.192,
     .  -2.040,  -1.961,  -1.111,  -1.111,  -1.209,  -1.209,  -1.382,  -1.382,
     .  -1.694,  -1.694,  -4.535,  -4.535,  -4.048,  -4.048,  14.456,  14.504,
     .  18.406,  17.916,  17.418,  16.966,  16.510,  16.055,  15.860,  15.374,
     .  14.897,  14.398,  13.983,  14.281,  14.238,  13.761,  13.926,  13.449/
      data (latitude(i),i=11761,11840)/
     .  13.057,  12.560,  12.888,  13.214,  13.556,  13.925,  13.430,  13.641,
     .  13.944,  13.650,  13.892,  14.384,  14.855,  14.774,  15.264,  15.753,
     .  16.243,  16.047,  16.547,  17.031,  17.527,  18.017,  18.509,  18.628,
     .  18.393,  18.379,  18.397,  20.833,  20.830,  20.480,  20.475,  19.574,
     .  19.571,  19.384,  19.378,  19.147,  19.147,  18.891,  18.891,  18.983,
     .  18.979,  15.047,  15.039,  14.221,  14.218,  14.059,  13.563,  14.024,
     .  14.058,  12.580,   2.321,  11.954,  12.147,  12.524,  12.576,  12.652,
     .  12.651,  13.143,   2.758,  13.092,  13.137,  12.476,  12.470,  12.602,
     .  12.599,  12.654,  12.166,  12.659,  12.653,  13.556,  13.553,  13.520,
     .  13.500,  13.287,  12.803,  12.312,  12.517,  12.995,  13.406,  13.525/
      data (latitude(i),i=11841,11920)/
     .  13.847,  13.845,  12.579,  12.535,  12.413,  11.924,  11.430,  11.282,
     .  11.757,  12.082,  12.533,  12.578,  11.536,  11.427,  10.965,  10.486,
     .  10.011,  10.416,  10.919,  11.303,  11.536,  10.427,   9.925,  10.423,
     .  10.427,  10.021,  10.021,   9.805,   9.525,   9.130,   8.633,   8.189,
     .   7.721,   7.221,   6.785,   6.299,   6.786,   7.231,   6.898,   6.415,
     .   5.915,   6.108,   5.951,   6.182,   6.646,   7.146,   7.640,   7.833,
     .   7.527,   7.780,   7.381,   6.932,   7.425,   7.903,   8.134,   8.472,
     .   8.687,   8.340,   8.535,   8.767,   9.061,   9.493,   9.809,   6.725,
     .   6.722,   6.378,   6.377,   6.093,   6.092,   5.338,   5.341,   7.036,
     .   7.036,   9.289,   9.288,   9.238,   9.238,  10.102,   9.614,   9.731/
      data (latitude(i),i=11921,12000)/
     .  10.120,  10.095,  11.277,  11.275,  11.215,  10.712,  10.238,   9.822,
     .  10.325,  10.761,  11.240,  11.215,  10.877,  10.403,   9.931,   9.432,
     .   9.364,   9.743,  10.064,  10.560,  10.974,  10.877,  10.731,  10.731,
     .  11.897,  11.635,  11.467,  10.985,  10.670,  10.426,  10.926,  11.423,
     .  11.907,  11.894,  12.323,  12.066,  12.270,  12.321,  11.956,  11.956,
     .  11.510,  11.510,  10.636,  10.629,   8.319,   8.318,   8.056,   8.050,
     .   8.121,   8.121,  11.409,  10.907,  10.414,  10.079,   9.892,   9.422,
     .   9.136,   8.757,   8.491,   8.954,   9.262,   9.613,  10.001,  10.348,
     .  10.759,  11.254,  11.402, -22.231, -22.231, -22.349, -22.349, -23.772,
     . -23.772, -25.798, -25.297, -24.801, -25.295, -25.779, -25.798, -27.342/
      data (latitude(i),i=12001,12080)/
     . -27.342, -27.717, -27.717, -38.773, -38.773, -39.499, -39.499, -38.561,
     . -38.561, -38.423, -38.423, -36.075, -36.045, -35.641, -35.759, -36.075,
     . -35.074, -35.074, -33.772, -33.772, -32.322. -32.322, -40.111, -39.609,
     . -40.109, -40.111, -40.267, -39.801, -40.156, -40.267, -40.474, -40.474,
     . -40.587, -40.587, -43.254, -43.254, -43.489, -43.489, -40.709, -40.709,
     . -40.469, -40.469, -40.592, -40.592, -42.731, -42.731, -26.099, -25.642,
     . -26.067, -26.099, -25.266, -25.266, -24.965, -24.965, -20.857, -20.857,
     . -15.439, -15.439, -15.309, -15.309, -15.022, -15.022, -14.611, -14.611,
     . -11.811, -11.358, -11.731, -11.831, -11.811, -11.928, -11.659, -11.174,
     . -11.354, -11.252, -11.740, -11.928, -11.314, -11.314, -11.715, -11.715/
      data (latitude(i),i=12081,12160)/
     . -11.670, -11.670, -11.424, -11.424, -12.057, -12.057, -13.522, -13.522,
     . -13.847, -13.847, -14.195, -13.747, -14.192, -14.239, -14.195, -14.893,
     . -14.893, -15.644, -15.644, -15.775, -15.775, -15.609, -15.609, -15.829,
     . -15.829. -16.740, -16.410, -16.757, -16.740, -17.130, -17.130, -10.179,
     . -10.179, -10.259, -10.259, -10.749, -10.749, -10.638, -10.638, -18.468,
     . -18.468, -19.165, -19.165, -20.143, -20.143, -20.324, -20.324, -20.527,
     . -20.527, -11.710, -11.710, -15.744, -15.744, -29.069, -29.069, -31.592,
     . -31.592, -41.689, -41.239, -40.984, -41.299, -40.817, -40.704, -41.162,
     . -41.634, -41.999, -42.477, -42.963, -43.115, -43.436, -43.719, -43.719,
     . -43.953, -44.179, -44.595, -44.898, -45.392, -45.797, -46.238, -46.284/
      data (latitude(i),i=12161,12240)/
     . -46.598, -46.639, -46.492, -46.116, -45.761, -45.266, -44.820, -44.787,
     . -44.787, -44.296, -44.024, -43.550, -43.319, -43.539, -43.859, -43.906,
     . -43.425, -43.358, -43.406, -43.006, -42.584, -42.178, -41.755, -41.687,
     . -47.221, -47.220, -46.793, -46.794, -46.722, -47.186, -46.688, -46.731,
     . -48.020, -48.023, -50.564, -50.568, -52.460, -52.466, -49.659, -49.669,
     . -47.680, -47.681, -44.315, -44.310, -44.014, -44.012, -45.630, -45.638,
     . -40.911, -40.918, -39.683, -40.182, -40.590, -41.029, -41.412, -41.388,
     . -40.883, -40.409, -39.935, -39.711, -39.461, -39.014, -38.693, -38.194,
     . -38.101, -36.803, -36.939, -36.708, -37.205, -37.683, -37.876, -37.986,
     . -37.658, -37.647, -38.138, -38.629, -39.088, -39.134, -39.590, -39.681/
      data (latitude(i),i=12241,12320)/
     . -38.101, -37.611, -37.135, -36.671, -36.171, -35.876, -36.330, -35.966,
     . -35.561, -35.087, -34.654, -35.019, -35.048, -35.365, -35.844, -36.325,
     . -36.827, -36.802, -36.350, -36.348, -36.830, -36.829, -31.439, -31.431,
     . -30.588, -30.575, -30.203, -30.205, -29.284, -29.285, -34.176, -34.176,
     . -46.804, -46.806,   5.297,   5.297,   7.512,   7.512,   6.722,   6.722,
     .   6.744,   6.744,   5.781,   5.781,   6.827,   6.827,   6.987,   6.987,
     .   5.282,   5.282,   7.327,   7.327,   7.424,   7.424,   7.349,   7.349,
     .   6.702,   6.702,   7.361,   7.361,   7.372,   7.372,   8.613,   8.613,
     .   8.976,   8.976,   8.577,   8.577,   7.506,   7.506,   7.378,   7.378,
     .   6.706,   6.706,  -0.495,  -0.495,   6.008,   5.799,   6.079,   6.079/
      data (latitude(i),i=12321,12400)/
     .   3.006,   3.006,   3.096,   3.096,   3.260,   3.260,   1.706,   1.706,
     .   1.534,   1.534,   1.846,   1.846,   1.346,   1.346,   1.018,   1.018,
     .   0.448,   0.448,   0.208,   0.208,  -0.599,  -0.599,  -0.673,  -0.673,
     .  -1.154,  -1.154,  -1.439,  -1.439,  -1.830,  -1.830,  -1.289,  -1.289,
     .  -1.333,  -1.333,  -5.680,  -5.680,  -6.264,  -6.264,  -6.092,  -6.092,
     .  -7.175,  -7.175,  -7.472,  -7.472,  -8.036,  -8.036,  -8.466,  -8.547,
     .  -4.501,  -4.501,  -4.439,  -4.439,  -3.119,  -3.119,  -2.764,  -2.764,
     .  -9.943,  -9.943,  -9.867,  -9.867,  -2.487,  -2.487,  -2.608,  -2.608,
     .   4.628,   4.571,   6.048,   6.062,  -5.658,  -5.658,   0.228,   0.228,
     .   0.809,   0.809,   5.633,   5.633,  24.309,  24.309,   7.135,   7.135/
      data (latitude(i),i=12401,12480)/
     .   8.911,   8.911,   7.612,   7.612,   7.329,   7.329,   7.464,   7.464,
     .   8.748,   8.748,  11.157,  11.157,  11.640,  11.640,  19.330,  19.330,
     .   7.767,   7.767,   9.334,   9.334,   9.219,   9.219,  11.706,  11.706,
     .  11.354,  11.354,  11.397,  11.397,   9.633,   9.633,   9.559,   9.559,
     .  10.451,  10.451,  11.232,  11.232,  14.581,  14.581,   7.027,   7.074,
     .   7.041,   6.972,   7.068,   7.108,   7.141,   7.066,   9.385,   9.333,
     .  -7.324,  -7.324,  -6.968,  -6.968,  -6.791,  -6.791,  -7.389,  -7.389,
     .  -6.604,  -6.852,  -7.214,  -7.061,  -6.693,  -6.604,  -7.575,  -7.575,
     .  -7.944,  -7.944,  -7.852,  -7.852,  -8.192,  -8.192,  -8.164,  -8.164,
     .  -8.414,  -8.414,  -8.710,  -8.710,  -8.701,  -8.701,  -8.509,  -8.509/
      data (latitude(i),i=12481,12560)/
     .  -7.971,  -8.356,  -8.321,  -7.971,  -8.378,  -8.378,  -7.535,  -7.756,
     .  -7.986,  -8.276,  -8.216,  -7.965,  -7.598,  -7.535,  -7.571,  -7.571,
     .  -7.468,  -7.468,  -9.041,  -9.041,  -8.997,  -8.997,  -9.111,  -9.111,
     .  -8.996,  -8.996,  -9.064,  -9.064,  -9.254,  -9.421,  -9.626,  -9.821,
     .  -9.661,  -9.254,  -8.310,  -8.768,  -9.160,  -8.977,  -8.483,  -8.310,
     .  -9.351,  -9.351,  -8.373,  -8.373,  -9.712,  -9.712, -10.206, -10.409,
     . -10.654, -10.691, -10.331, -10.206, -11.290, -11.290, -11.472, -11.709,
     . -11.546, -11.472, -10.754, -10.754, -11.282, -11.282, -11.602, -11.602,
     .  -5.300,  -5.300,  -5.414,  -5.414,  -5.467,  -5.467,  -5.490,  -5.490,
     . -19.248, -19.248, -19.680, -19.680, -20.137, -20.137, -20.700, -20.396/
      data (latitude(i),i=12561,12640)/
     . -20.167, -20.630, -20.997, -21.362, -21.665, -21.928, -22.178, -22.374,
     . -21.939, -21.611, -21.329, -20.973, -20.700, -22.571, -22.571, -20.659,
     . -20.659, -20.723, -20.723, -20.953, -20.953, -21.558, -21.558, -22.331,
     . -22.331, -20.196, -20.196, -19.588, -19.588, -18.854, -18.854, -17.660,
     . -17.660, -16.818, -16.818, -16.198, -16.198, -15.888, -15.888, -15.303,
     . -15.303, -15.479, -15.479, -14.323, -14.323, -13.860, -13.860, -13.686,
     . -13.686, -13.554, -13.554, -13.434, -13.434, -13.175, -13.175, -14.999,
     . -15.494, -15.519, -15.036, -14.999, -15.610, -15.610, -15.756, -15.756,
     . -16.450, -16.013, -16.259, -16.450, -12.518, -12.518, -16.528, -16.539,
     . -16.494, -16.528, -17.003, -16.538, -16.403, -16.202, -16.155, -16.174/
      data (latitude(i),i=12641,12720)
     . -16.525, -16.743, -17.003, -16.833, -16.833, -17.350, -17.350, -17.813,
     . -17.813, -18.105, -18.105, -17.688, -17.688, -18.264, -18.162, -17.667,
     . -17.373, -17.434, -17.854, -18.190, -18.264, -16.719, -16.719, -17.101,
     . -17.101, -17.284, -17.284, -18.496, -18.496, -18.374, -18.374, -18.928,
     . -18.928, -19.000, -19.000, -19.197, -19.197, -18.967, -18.967, -18.552,
     . -18.552, -17.775, -17.775, -17.250, -17.250, -17.431, -17.431, -17.177,
     . -17.177, -17.963, -17.963, -18.191, -18.191, -18.641, -18.641, -18.966,
     . -18.966, -19.129, -19.129, -19.159, -19.159, -19.844, -19.844, -20.671,
     . -20.671, -16.999, -16.786, -16.968, -16.999, -21.704, -21.704, -17.987,
     . -17.987, -24.328, -24.327, -24.664, -24.663,   7.714,   7.714,   9.576/
      data (latitude(i),i=12721,12800)/
     .   9.576,  13.635,  13.635,  14.178,  14.178,  15.069,  15.069,  15.249,
     .  15.249,  16.379,  16.379,  16.706,  16.706,  17.313,  17.313,  17.606,
     .  17.606,  18.173,  18.173,  18.802,  18.802,  19.655,  19.655,  20.520,
     .  20.520,   5.315,   5.315,   6.936,   6.936,   7.050,   7.050,   7.313,
     .   7.313, -13.343, -13.343, -13.805, -13.805, -14.053, -14.053, -14.308,
     . -14.308, -21.268, -21.268, -19.101, -19.101,  -9.210,  -9.210, -14.369,
     . -14.369, -14.273, -14.273, -15.624, -15.624, -18.667, -18.667, -18.667,
     . -18.667, -19.786, -19.786, -19.735, -19.735, -21.446, -21.446,  -8.985,
     .  -8.985, -21.230, -21.230, -21.912, -21.912,  -8.582,  -8.582,  -9.372,
     .  -9.372, -14.288, -14.288, -14.185, -14.185, -18.826, -18.826, -18.705/
      data (latitude(i),i=12801,12880)/
     . -18.705, -19.652, -19.652, -10.890, -10.890, -11.548, -11.548, -10.037,
     . -10.037, -10.430, -10.430, -10.386, -10.386, -18.081, -18.081, -18.880,
     . -18.880, -19.272, -19.272, -19.836, -19.836, -20.019, -20.019, -19.845,
     . -19.845, -20.184, -20.184, -16.674, -16.674, -16.904, -16.904, -27.556,
     . -27.556, -25.076, -25.076, -22.655, -22.655, -11.464, -11.464, -27.914,
     . -27.914, -10.442, -10.442,  -9.899,  -9.899,  -9.723,  -9.723,  -9.329,
     .  -9.329,  -8.885,  -8.885,  -8.806,  -8.806,  -7.973,  -7.973,  -9.896,
     .  -9.896,  -9.930,  -9.930, -10.094, -10.094,  -5.618,  -5.618,  -4.055,
     .  -4.055,  -0.385,  -0.385,   1.717,   1.717,   3.799,   3.799,   4.706,
     .   4.706, -23.139, -23.139, -21.709, -21.709, -15.749, -15.749, -16.165/
      data (latitude(i),i=12881,12960)/
     . -16.165, -17.735, -17.735, -17.572, -17.572, -16.826, -16.826, -16.770,
     . -16.770, -16.531, -16.531, -15.844, -15.844, -16.973, -16.973, -16.977,
     . -16.977, -17.661, -17.661, -22.514, -22.514, -23.401, -23.401, -23.873,
     . -23.873, -21.534, -21.487, -21.568, -21.531, -22.029, -22.019, -21.466,
     . -21.466, -21.358, -21.342, -18.571, -18.485, -18.362, -18.269, -17.318,
     . -17.309, -17.318, -17.358, -18.774, -18.754, -20.880, -20.788, -21.873,
     . -21.821, -21.871, -21.876, -19.656, -19.612, -19.164, -19.138, -19.219,
     . -19.202, -18.419, -18.340, -18.220, -18.066, -17.810, -17.689, -17.703,
     . -17.786, -15.977, -15.981, -15.799, -15.831, -16.018, -16.110, -16.145,
     . -16.186, -16.196, -16.231, -17.622, -17.537, -16.994, -16.974, -17.457/
      data (latitude(i),i=12961,13040)/
     . -17.434, -16.672, -16.627, -16.638, -16.581, -16.545, -16.455, -16.743,
     . -16.712, -16.673, -16.645, -16.483, -16.417, -16.347, -16.325, -16.074,
     . -16.085, -16.149, -16.194, -15.950, -15.891, -15.908, -15.758, -15.480,
     . -15.442, -14.512, -14.446, -14.624, -14.603, -14.621, -14.666, -14.347,
     . -14.367, -14.383, -14.423, -14.438, -14.458, -14.461, -14.430, -14.500,
     . -14.436, -15.318, -15.378, -15.481, -15.536, -15.838, -15.883, -16.064,
     . -16.054, -16.086, -16.350, -16.408, -16.460, -16.727, -16.755, -17.491,
     . -17.419, -17.406, -17.353, -17.386, -17.326, -19.931, -19.897, -15.825,
     . -15.776, -15.270, -15.218, -15.304, -15.291, -15.265, -15.246, -15.233,
     . -15.206, -15.191, -15.142, -15.021, -14.988, -15.002, -14.922, -14.944/
      data (latitude(i),i=13041,13120)/
     . -14.978, -14.900, -14.856, -15.043, -15.069, -16.482, -16.441, -16.453,
     . -16.419, -16.420, -16.461, -16.829, -16.774, -15.798, -15.765, -15.813,
     . -15.792, -21.343, -21.324, -21.317, -21.301, -19.282, -19.277, -19.355,
     . -19.331, -20.800, -20.799, -17.382, -17.349, -16.840, -16.831, -14.183,
     . -14.165, -14.185, -14.157, -18.131, -18.115, -17.970, -17.952, -17.612,
     . -17.625, -17.836, -17.815, -14.427, -14.438, -15.266, -15.287, -14.908,
     . -14.899, -15.057, -15.084  -14.910, -14.936, -14.902, -14.873, -15.202,
     . -15.226, -19.874, -19.857, -20.433, -20.388, -54.752, -54.752,  28.155,
     .  27.833,  27.678,  27.326,  27.073,  26.771,  26.700,  26.506,  26.722,
     .  26.995,  27.171,  27.118,  27.139,  26.949,  26.697,  26.976,  26.950/

      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c---------------------------------------------------------------------------------
      subroutine MapIt
c---------------------------------------------------------------------------------
c     Draw an Earth map and overlay a flight path onto it
c
c     developer:    David F. Smith
c     date:             February 1991
c
c.....common block definition files

      include 'CrvDat.inc'
      include 'FntCom.inc'
      include 'LatCom.inc'
      include 'LngCom.inc'
      include 'MapMenu.inc'
      include 'OptFlg.inc'
      include 'PenCom.inc'
      include 'MapCom.inc'
      include 'PntAbs.inc'
      include 'TicDat.inc'
      include 'VuWind.inc'
      include 'WinLim.inc'

c.....graphics window records

      common / MapWindow /        MapWPtr
      record / WindowPtr /        MapWptr

c.....set up pointer for QuickDraw globals

      common  / QDGPtr    /       QDG
      pointer / QDGlobals /       QDG

c.....Picture record handle and pointer

      common / pict      /        PictHndl
      record / PicHandle /        PictHndl

c.....pointer to off screen bit map

      record / GrafPtr   /        OffScreen

c.....Handle to off screen bit map memory contents

      record / Handle    /        myBitH

c.....event record

      record / EventRecord /      TheEvent

c.....cursor handle

      record / CursHandle /       CursorHndl

c.....user option code

```

```
      integer*2                    Option

c.....continue reading data flag

      integer*2                    goflag

c.....define logical*1 to represent Boolean argument from Event monitor

      logical*1                    AnEvent

c.....define an initialization flag

      logical*1                    FirstTime

c-------------------------------------------------------------------------------

c.....set default graph window limits

      iGxMin = QDG^.screenBits.bounds.left
      iGxMax = QDG^.screenBits.bounds.right
      iGyMin = QDG^.screenBits.bounds.top
      iGyMax = QDG^.screenBits.bounds.bottom - 38

c.....get screen resolution

      call ScreenRes ( %ref(iHRes) , %ref(iVRes) )
      MapHRes = float ( iHRes )
      MapVRes = float ( iVRes )

c.....get default graph window size

      DefWidth  = float ( iGxMax - iGxMin )/MapHRes
      DefHeight = float ( iGyMax - iGyMin )/MapVRes

c.....initialize window size to the default

      MapWidth  = DefWidth
      MapHeight = DefHeight

c.....set up the 'Map' menu

c      call SetUpMapMenu
c      call UnloadSeg ( %loc(SetUpMapMenu) )

c------------------------------------------------------------------------------
c.....the main execution loop begins here
c------------------------------------------------------------------------------

      Option = oNew
      FirstTime = .true.

      do while ( Option.eq.oNew .or. Option.eq.oRedraw )

c........set dialog font to Chicago ( system )

         FntNam = 'Chicago'
         call GetFNum ( %val(FntNam) , FntNum )
         call setDAfont ( %val(FntNum) )

c........show 'Map' options menu

c         call InsertMenu ( %val(MapMenuHndl) , %val(0) )
c         call DrawMenuBar

```

```
c........initialize user option to Cycle (after the first pass)

         if(FirstTime) then
           FirstTime = .false.
         else
           Option = oCycle
         end if

c........monitor and respond to events

         do while ( Option.eq.oCycle )
            AnEvent = GetNextEvent ( %val(EveryEvent) , %ref(TheEvent) )
            if ( AnEvent ) then
               call EventHandler ( TheEvent , Option )
            end if
         end do

c........clear and redraw the menu bar

c         call ClearMenuBar
c         call DrawMenuBar

c........set up the map appearance via a dialog window

         if ( Option.eq.oNew ) then
            call SetUpTheMap ( Option )
         end if

c........draw the map via QuickDraw

         if ( Option.eq.oNew ) then
            call DrawTheMap  ( Option )
            call OpenBitMap  ( offScreen , myBitH , MapWPtr.WP^.portRect )
            call SetPort     ( %val(offScreen) )
            call CopyBits    ( %ref(MapWPtr.WP^.portBits) ,
     &                         %ref(OffScreen.GrafP^.portBits) ,
     &                         %ref(MapWPtr.WP^.portRect) ,
     &                         %ref(OffScreen.GrafP^.portRect) ,
     &                         %val(srcCopy) , %val(nil) )
         else if ( Option.eq.oReDraw ) then
            call SetPort     ( %val(MapWPtr) )
            call BeginUpdate ( %val(MapWPtr) )
            call CopyBits    ( %ref(OffScreen.GrafP^.portBits) ,
     &                         %ref(MapWPtr.WP^.portBits) ,
     &                         %ref(OffScreen.GrafP^.portRect) ,
     &                         %ref(MapWPtr.WP^.portRect) ,
     &                         %val(srcCopy) , %val(nil) )
            call EndUpdate   ( %val(MapWPtr) )
            call SelectWindow( %val(MapWPtr) )
         end if

c........enable the save and redraw menu items after first Map is complete

         if ( iMadeFirstMap.eq.0 ) then
            call MapMenuSet ( itemSaveMap , enableTheItem )
            call MapMenuSet ( itemRedraw  , enableTheItem )
            iMadeFirstMap = 1
         end if

      end do

c.....eliminate the Map window

      call HideWindow ( %val(MapWptr) )
      call DisposeWindow ( %val(MapWptr) )

c.....get rid of 'Map' menu

c      call DeleteMenu ( %val(MapMenuID) )
c      call DisposeMenu ( %val(MapMenuHndl ) )
c      call ClearMenuBar
c      call DrawMenuBar

c.....return to calling routine

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-----------------------------------------------------------------------------------
      subroutine MapMenuSet ( item , enable )
c-----------------------------------------------------------------------------------
c     This routine lets other routines enable or disable menu items without
c     the calling routines having to know about menu structures.

      logical*2                    enable
      integer*2                    item

c.....Options menu file

      include 'MapMenu.inc'

c.....either enable or disable designated item

      if ( enable ) then
         call EnableItem ( %val(MapMenuHndl) , %val(item) )
      else
         call DisableItem ( %val(MapMenuHndl) , %val(item) )
      end if

c.....redraw the menu bar

      call DrawMenuBar

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main
```

```

c-------------------------------------------------------------------------------
Segment Main
      subroutine MenuSet ( menuID, menuItem, enable )
c-------------------------------------------------------------------------------
c     This routine lets other routines enable or disable menu items without
c     the calling routines having to know about menu structures.

!!SETC USINGINCLUDES = FALSE

      implicit none

c.....declare Boolean flag

      logical*1 enable

c.....accept the input arguments

      integer*2 menuID, menuItem

c.....declare a structure for getting the menu handle

      record / MenuHandle / menu

c-------------------------------------------------------------------------------

c.....get the menu's handle

      menu.menuH = GetMHandle ( %val( menuID ))

      if (enable) then
         call EnableItem  ( %val( menu ), %val( menuItem ))
      else
         call DisableItem ( %val( menu ), %val( menuItem ))
      endif

c.....display the results

      call DrawMenuBar

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-------------------------------------------------------------------------------
      subroutine MovAbs ( ix , iy )
c-------------------------------------------------------------------------------
c     Move to absolute graphics window position .

      include 'PntAbs.inc'
      include 'WinLim.inc'

      ixabs  = ix
      iyabs  = iy

      call MoveTo ( %val(ixabs) , %val(iGyMax-iyabs) )

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-------------------------------------------------------------------------------
      subroutine MovRel ( ix , iy )
c-------------------------------------------------------------------------------
c     Move to relative graphics window position .

      include 'PntAbs.inc'
      include 'WinLim.inc'

      ixabs  = ixabs + ix
      iyabs  = iyabs + iy

      call MoveTo ( %val(ixabs) , %val(iGyMax-iyabs) )

      return
      end
```

```
      SUBROUTINE NOGAPS( Z, X, Y, TBLV1, TBLV2, FIRST )
C***************************************************************************C
C                                                                           C
C     THIS PROGRAM DOES A TABLE LOOKUP FOR EAST AND NORTH WIND VECTORS      C
C     AT THE ALTITUDE, LATITUDE,AND LONGITUDE SPECIFIED.                    C
C                                                                           C
C     THE OUTPUTS OF THIS PROGRAM ARE EAST AND NORTH WIND VECTORS.          C
C                                                                           C
C***************************************************************************C
C
      LOGICAL FIRST
      REAL F,V
      INTEGER I,IP1,NL,NU,NGRD,MGRD,MAT
      DIMENSION  F    ( 3000 ),  V    (  3 ),  DDV (  3 )
      DIMENSION  I    (  3 ),  IP1 (  3 ),  NL  (  3 ),  NU  (  3 )
C
      SAVE
C
C***************************************************************************C
C
      V(1)  = Z
      V(2)  = Y
      V(3)  = X
C
      I(1)  = 1
      I(2)  = 1
      I(3)  = 1
C
C     READ IN THE EAST VECTOR FILE
C
      IF(FIRST)THEN
      OPEN(UNIT=15,STATUS='OLD',FILE='EAST.DAT')
C
```

```
      J=1
      READ(15,1)F(J),F(J+1),F(J+2)
 1    FORMAT(F8.3,F8.3,F8.3)
      READ(15,*)
      MGRD=INT(F(2))
      NGRD=INT(F(3))
      MAT=MGRD*NGRD
 C
      DO J=4,12,4
      READ(15,6)F(J),F(J+1),F(J+2),F(J+3)
      END DO
      READ(15,*)
      DO J=13,MGRD+12,4
      READ(15,6)F(J),F(J+1),F(J+2),F(J+3)
      END DO
      READ(15,*)
      DO J=MGRD+13,MGRD+NGRD+12,4
      READ(15,6)F(J),F(J+1),F(J+2),F(J+3)
      END DO
      READ(15,*)
      DO J=MGRD+NGRD+13,MGRD+NGRD+MAT+12,4
      READ(15,5)F(J),F(J+1),F(J+2),F(J+3)
      END DO
      DO JJ=1,8
      DO J=(JJ-1)*MAT+(MGRD+NGRD+MAT+13),JJ*MAT+(MGRD+NGRD+MAT+12),4
      READ(15,5)F(J),F(J+1),F(J+2),F(J+3)
      END DO
      END DO
 C
 5    FORMAT(F8.3,F8.3,F8.3,F8.3)
 6    FORMAT(F9.3,F9.3,F9.3,F9.3)

      CLOSE(15)
        ENDIF
 C
 C.....BEGIN TABLE LOOKUP FOR EAST VECTOR
 C
      N1 = INT(F(1))
      N2 = INT(F(2))
      N3 = INT(F(3))
 C
 C.....COMPUTE UPPER BOUNDS ON INDICES FOR Z, Y, AND X
 C
      NU(1) = 3 + N1
      NU(2) = NU(1) + N2
      NU(3) = NU(2) + N3
 C
 C.....COMPUTE LOWER BOUNDS ON INDICES FOR Z, Y, AND X
 C
      NL(1) = 4
      NL(2) = NL(1) + N1
      NL(3) = NL(2) + N2
 C
 C.....LOOP FOR ALL THREE INDICES
 C
      DO 200 J=1,3
 C
 C.....GET INDICES IN BOUNDS
 C
      IF( I(J) .LT. NL(J)    ) I(J) = NL(J)
      IF( I(J) .GT. NU(J)-1 ) I(J) = NU(J) - 1
 C
 C.....FIND GREATEST LOWER BOUND ON INDEX
 C
```

```
110     CONTINUE
        IP1(J) = I(J) + 1
        IF( V(J)   .LE. F(IP1(J)) ) GO TO 130
        IF( IP1(J) .EQ.     NU(J) ) GO TO 150
        I(J) = IP1(J)
        GO TO 110
120     CONTINUE
        IF( I(J) .EQ. NL(J) ) GO TO 140
        I(J) = I(J) - 1
130     CONTINUE
        IF( V(J) .LT. F(I(J)) ) GO TO 120
140     CONTINUE
        IP1(J) = I(J) + 1
150     CONTINUE
C
C....... GET PARTIALS OF INDEPENDENT VARIABLES
C
        DDV(J) = ( V(J) - F(I(J)) )/( F(IP1(J)) - F(I(J)) )
C
200     CONTINUE
C
C....... GET ALL INDICES INTO COEFFICIENT ARRAY
C
        N2 = N1*N2
C
        NN = N1*( I(2) - NL(2) ) + N2*( I(3) - NL(3) )
C
        N111 = NU(3) + 1 + I(1) - NL(1) + NN
        N211 = N111 +  1
        N121 = N111 + N1
        N221 = N121 +  1
        N112 = N111 + N2
        N212 = N112 +  1
        N122 = N121 + N2
        N222 = N122 +  1
C
        P1A1 = DDV(1)*( F(N211) - F(N111) ) + F(N111)
        P1B1 = DDV(1)*( F(N221) - F(N121) ) + F(N121)
        P2A1 = DDV(1)*( F(N212) - F(N112) ) + F(N112)
        P2B1 = DDV(1)*( F(N222) - F(N122) ) + F(N122)
C
        T11  = DDV(2)*( P1B1 - P1A1 ) + P1A1
        T21  = DDV(2)*( P2B1 - P2A1 ) + P2A1
C
C.....TBLV1 IS THE EAST VECTOR OUTPUT
C
        TBLV1 = DDV(3)*( T21  - T11  ) + T11
C
C.....READ IN THE NORTH VECTOR
C
        IF(FIRST)THEN
        OPEN(UNIT=15,STATUS='OLD',FILE='NORTH.DAT')
        J=1
        READ(15,101)F(J),F(J+1),F(J+2)
101     FORMAT(F8.3,F8.3,F8.3)
        READ(15,*)
        MGRD=INT(F(2))
        NGRD=INT(F(3))
        MAT=MGRD*NGRD
C
        DO J=4,12,4
        READ(15,106)F(J),F(J+1),F(J+2),F(J+3)
        END DO
        READ(15,*)
```

```
      DO J=13,MGRD+12,4
      READ(15,106)F(J),F(J+1),F(J+2),F(J+3)
      END DO
      READ(15,*)
      DO J=MGRD+13,MGRD+NGRD+12,4
      READ(15,106)F(J),F(J+1),F(J+2),F(J+3)
      END DO
      READ(15,*)
      DO J=MGRD+NGRD+13,MGRD+NGRD+MAT+12,4
      READ(15,105)F(J),F(J+1),F(J+2),F(J+3)
      END DO
      DO JJ=1,8
      DO J=(JJ-1)*MAT+(MGRD+NGRD+MAT+13),JJ*MAT+(MGRD+NGRD+MAT+12),4
      READ(15,105)F(J),F(J+1),F(J+2),F(J+3)
      END DO
      END DO
C
105   FORMAT(F8.3,F8.3,F8.3,F8.3)
106   FORMAT(F9.3,F9.3,F9.3,F9.3)

      FIRST = .FALSE.
      CLOSE(15)
        ENDIF
C
C.....BEGIN TABLE LOOKUP FOR NORTH VECTOR
C
      P1A1 = DDV(1)*( F(N211) - F(N111) ) + F(N111)
      P1B1 = DDV(1)*( F(N221) - F(N121) ) + F(N121)
      P2A1 = DDV(1)*( F(N212) - F(N112) ) + F(N112)
      P2B1 = DDV(1)*( F(N222) - F(N122) ) + F(N122)
C
      T11  = DDV(2)*( P1B1 - P1A1 ) + P1A1
      T21  = DDV(2)*( P2B1 - P2A1 ) + P2A1
C
C.....TBLV2 IS THE NORTH VECTOR OUTPUT
C
      TBLV2 = DDV(3)*( T21  - T11  ) + T11
C
      IZ = I(1)
      IY = I(2)
      IX = I(3)
C
      RETURN
      END
```

```
c-------------------------------------------------------------------------------
      integer*2 function nquant ( value , quantm )
c-------------------------------------------------------------------------------
c     quantize the input value to the nearest number of counts in either the
c     positive or negative direction as indicated by the sign of the
c     quantization factor

c.....halt if quantization factor is zero

      if ( quantm.eq.0.0 ) then
         pause 'quantization factor cannot be zero'
         call exit
      end if

c.....quantize to the nearest quantum in the positive direction

      if ( quantm.gt.0.0 ) then
         nquant = inint ( value/quantm + 0.5 )
```

```
            remain = quantm*float(nquant) - value
            if ( remain.eq.quantm ) then
               nquant = nquant - 1
            end if
         end if

c.....quantize to the nearest quantum in the negative direction

         if ( quantm.lt.0.0 ) then
            quantp = - quantm
            nquant = inint ( value/quantp - 0.5 )
            remain = value - quantp*float(nquant)
            if ( remain.eq.quantp ) then
               nquant = nquant + 1
            end if
         end if

         return
         end
```

```
c-------------------------------- --- -----------------------------------------
         integer*2 function ntrvl ( x , xt , nx , idir )
c------------------------------------------------------------------------------
c        determine the index in a monotonic data table associated with the input
c        value.  Uses binary search algorithm.

         integer*2           nx
         integer*2           idir
         integer*2           ixmid
         integer*2           ixmin
         integer*2           ixmax
         real*4              xt(nx)

c.....the data table has ascending values

         if ( idir.gt.0 ) then
            if ( x.ge.xt(1) .and. x.le.xt(nx) ) then
               ixmin  = 1
               ixmax  = nx
               do while ( ixmax.ne.ixmin+1 )
                  ixmid  = ( ixmin + ixmax )/2
                  if ( x.ge.xt(ixmid) ) then
                     ixmin  = ixmid
                  else
                     ixmax  = ixmid
                  end if
               end do
               ntrvl  = ixmin
               return
            else if ( x.lt.xt(1) ) then
               ntrvl  = 0
               return
            else if ( x.gt.xt(nx) ) then
               ntrvl  = nx
               return
            end if

c.....the data has descending values

         else if ( idir.lt.0 ) then
            if ( x.le.xt(1) .and. x.ge.xt(nx) ) then
               ixmin  = 1
               ixmax  = nx
            do while ( ixmax.ne.ixmin+1 )
               ixmid  = ( ixmin + ixmax )/2
               if ( x.le.xt(ixmid) ) then
                  ixmin  = ixmid
               else
                  ixmax  = ixmid
               end if
            end do
            ntrvl  = ixmin
            return
         else if ( x.gt.xt(1) ) then
            ntrvl  = 0
            return
         else if ( x.lt.xt(nx) ) then
            ntrvl  = nx
            return
         end if

c.....the data is not monotonic

      else if ( idir.eq.0 ) then
      end if

      return
      end
```

```
c-------------------------------------------------------------------------------
      integer function NumChr ( ChrStr , lngth )
c-------------------------------------------------------------------------------
c     left justify the Character string [ChrStr] and determine the number of non
c     blank characters in it.

      character*(*) ChrStr

      ileft  = 1
      iright = 1

c.....get leftmost non blank character

      do i = 1 , lngth
         if ( ChrStr(i:i).ne.' ' ) then
            ileft  = i
            leave
         end if
      end do

c.....get rightmost non blank character

      do i = lngth , 1 , -1
         if ( ChrStr(i:i).ne.' ' ) then
            iright = i
            leave
         else
            if ( i.eq.1 ) then
               NumChr = 0
               return
            end if
         end if
      end do

c.....get number of non blank characters

      NumChr = iright - ileft + 1
```

```
c.....left justify the string

      ChrStr(1:NumChr)        = ChrStr(ileft:iright)
      ChrStr(NumChr+1:lngth) = ' '

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-----------------------------------------------------------------------
      subroutine OpenBitMap ( newOffScreen , myBitH , inBounds )
c-----------------------------------------------------------------------
c     open an off screen bit map to save the contents of the graphics window

      record / GrafPtr /          savePort
      record / GrafPtr /          newPort
      record / GrafPtr /          newOffScreen
      record / GrafPort /         myNewPort
      record / Handle /           myBitH
      record / Rect /             inBounds
      integer*4                   mySize

c.....get a pointer to the current port

      call GetPort ( %ref(savePort) )

c.....open a new port

      mySize = jSizeOf ( myNewPort )
      newPort = NewPtr ( %val(mySize) )
      call OpenPort ( %val(newPort) )

c.....set port attributes and allocate a locked memory block

      newPort.GrafP^.portRect           = inBounds
      call RectRgn ( %val(newPort.GrafP^.clipRgn) , %val(inBounds) )
      call RectRgn ( %val(newPort.GrafP^.visRgn)  , %val(inBounds) )
      newPort.GrafP^.portBits.bounds    = inBounds
      newPort.GrafP^.portBits.rowBytes = ( inBounds.right - inBounds.left + 15 )/16*2
      mySize                            = newPort.GrafP^.portBits.rowBytes*(
inBounds.bottom - inBounds.top )
      myBitH                            = NewHandle ( %val(mySize) )
      call HLock ( %val(myBitH) )
      newPort.GrafP^.portBits.baseAddr = myBitH.bhdl^.bptr

c.....erase the port since it is just memory

      call EraseRect ( %val(inBounds) )

c.....save a pointer to the new off screen bit map and restore the previous
c     window

      newOffScreen = newPort
      call SetPort ( %val(savePort) )
      return
      end


!!s PenDat
c-----------------------------------------------------------------------
      block data PenDat
c-----------------------------------------------------------------------
c     array of pen commands
c        0 Ÿ move Ÿ pen up
c        1 Ÿ draw Ÿ pen down
c        2 Ÿ end of data

      include 'PenCom.inc'

      data (PenCommand(i),i=    1,   80)/
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i=   81,  160)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i=  161,  240)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      . 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      . 1      , 1      , 0      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i=  241,  320)/
     . 1      , 1      . 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1
      data (PenCommand(i),i=  321,  400)/
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i=  401,  480)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i=  481,  560)/
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      /
      data (PenCommand(i),i=  561,  640)/
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i=  641,  720)/
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      /
      data (PenCommand(i),i=  721,  800)/
     . 1      , 0      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 0      , 1
      data (PenCommand(i),i=  801,  880)/
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i=  881,  960)/
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , -     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     /
      data (PenCommand(i),i=  961, 1040)/
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     /
      data (PenCommand(i),i= 1041, 1120)/
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 1121, 1200)/
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 0     , 1     /
      data (PenCommand(i),i= 1201, 1280)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , [illegible] , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 0     /
      data (PenCommand(i),i= 1281, 1360)/
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 1     , 1     , 0     , 1     , 0     /
      data (PenCommand(i),i= 1361, 1440)/
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     /
      data (PenCommand(i),i= 1441, 1520)/
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 1     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 1     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     /
      data (PenCommand(i),i= 1521, 1600)/
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     /
      data (PenCommand(i),i= 1601, 1680)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 1681, 1760)/
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i= 1761, 1840)/
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      /
      data (PenCommand(i),i= 1841, 1920)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      /
      data (PenCommand(i),i= 1921, 2000)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i= 2001, 2080)/
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1        1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i= 2081, 2160)/
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i= 2161, 2240)/
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     /
      data (PenCommand(i),i= 2241, 2320)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 1     , 1     , 0     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     /
      data (PenCommand(i),i= 2321, 2400)/
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 2401, 2480)/
     . 0     , 1     , 0     , 1     , 1     , 1     , 0     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 1     , 0     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     /
      data (PenCommand(i),i= 2481, 2560)/
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 2561, 2640)/
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
      data (PenCommand(i),i= 2641, 2720)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 0      , 1      , 1      /
      data (PenCommand(i),i= 2721, 2800)/
     . 1      , 0      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 0      , 1      , 0      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 0      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 0      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      /
      data (PenCommand(i),i= 2801, 2880)/
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 0      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i= 2881, 2960)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      /
      data (PenCommand(i),i= 2961, 3040)/
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 1      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 1      , 0      , 1      , 0      , 1      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 0      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      /
      data (PenCommand(i),i= 3041, 3120)/
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 1      , 1      , 0      , 1      , 0
      data (PenCommand(i),i= 3121, 3200)
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     /
      data (PenCommand(i),i= 3201, 3280)/
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     /
      data (PenCommand(i),i= 3281, 3360)/
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 3361, 3440)/
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 0     , 1     , 1     /
      data (PenCommand(i),i= 3441, 3520)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     /
      data (PenCommand(i),i= 3521, 3600)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     /
      data (PenCommand(i),i= 3601, 3680)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 3681, 3760)/
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 3761, 3840)/
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 3841, 3920)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     /
      data (PenCommand(i),i= 3921, 4000)/
     . 1     , 0     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     /
      data (PenCommand(i),i= 4001, 4080)/
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 1     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 1     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     /
```

```
 data (PenCommand(i),i= 4081, 4160)/
. 0       , 1       , 0       , 1       , 0       , 1       , 0       , 1       ,
. 1       , 1       , 0       , 1       , 0       , 1       , 0       , 1       ,
. 0       , 1       , 0       , 1       , 0       , 1       , 0       , 1       ,
. 0       , 1       , 0       , 1       , 0       , 1       , 0       , 1       ,
. 0       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
. 0       , 1       , 0       , 1       , 0       , 1       , 0       , 1       ,
. 0       , 1       , 0       , 1       , 1       , 1       , 1       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 0       , 1       ,
. 0       , 1       , 0       , 1       , 0       , 1       , 0       , 1       ,
. 0       , 1       , 0       , 1       , 0       , 1       , 0       , 1       /
 data (PenCommand(i),i= 4161, 4240)/
. 0       , 1       , 1       , 1       , 0       , 1       , 0       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 0       , 1       , 0       ,
. 1       , 0       , 1       , 0       , 1       , 1       , 1       , 1       ,
. 0       , 1       , 0       , 1       , 0       , 1       , 0       , 1       ,
. 0       , 1       , 1       , 1       , 0       , 1       , 0       , 1       ,
. 0       , 1       , 0       , 1       , 0       , 1       , 0       , 1       ,
. 0       , 1       , 1       , 1       , 0       , 1       , 1       , 1       ,
. 1       , 1       , 1       , 0       , 1       , 0       , 1       , 0       ,
. 1       , 0       , 1       , 0       , 1       , 0       , 1       , 0       /
 data (PenCommand(i),i= 4241, 4320)/
. 1       , 0       , 1       , 0       , 1       , 1       , 1       , 1       ,
. 1       , 0       , 1       , 0       , 1       , 0       , 1       , 0       ,
. 1       , 0       , 1       , 0       , 1       , 0       , 1       , 0       ,
. 1       , 0       , 1       , 0       , 1       , 0       , 1       , 0       ,
. 1       , 1       , 1       , 1       , 1       , 0       , 1       , 1       ,
. 1       , 1       , 1       , 1       , 0       , 1       , 0       , 1       ,
. 0       , 1       , 0       , 1       , 0       , 1       , 0       , 1       ,
. 0       , 1       , 0       , 1       , 0       , 1       , 0       , 1       ,
. 0       , 1       , 0       , 1       , 0       , 1       , 0       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 0       , 1       , 0       /
 data (PenCommand(i),i= 4321, 4400)/
. 1       , 0       , 1       , 1       , 1       , 1       , 1       , 1       ,
. 0       , 1       , 1       , 1       , 1       , 1       , 1       , 0       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 0       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
. 1       , 1       , 0       , 1       , 1       , 1       , 1       , 1       ,
. 1       , 1       , 1       , 0       , 1       , 1       , 1       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 1       , 0       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
. 0       , 1       , 1       , 0       , 1       , 1       , 1       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       /
 data (PenCommand(i),i= 4401, 4480)/
. 1       , 0       , 1       , 1       , 1       , 1       , 0       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
. 1       , 0       , 1       , 1       , 1       , 1       , 1       , 0       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 1       , 0       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
. 1       , 1       , 1       , 0       , 1       , 1       , 1       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
. 1       , 1       , 0       , 1       , 1       , 1       , 1       , 1
 data (PenCommand(i),i= 4481, 4560)/
. 1       , 1       , 0       , 1       , 1       , 1       , 1       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
. 1       , 1       , 1       , 1       , 0       , 1       , 0       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
. 1       , 0       , 1       , 1       , 1       , 1       , 1       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 0       , 1       , 1       ,
. 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
. 1       , 0       , 1       , 1       , 0       , 1       , 1       , 0       ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 4561, 4640)/
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 4641, 4720)/
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 4721, 4800)/
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     . 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 4801, 4880)/
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 0     , 1     /
      data (PenCommand(i),i= 4881, 4960)/
     . 0     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 1     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 0     , 1     ,
      data (PenCommand(i),i= 4961, 5040)
     . 1     , 1     , 1     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     . 1     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1    , 1    , 1     1    , 0    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 0    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    . 1    ,
     . 1    , 1    , 1    , 1    , 0    , 1    . 1    ,      /
      data (PenCommand(i),i= 5041, 5120)/
     . 0    , 1    , 1    , 1    , 0    , 1    , 0    , 1    ,
     . 0    , 1    , 0    , 1    , 0    , 1    , 0    , 1    ,
     . 0    , 1    , 0    , 1    , 0    , 1    , 0    , 1    ,
     . 0    , 1    , 1    , 1    , 1    , 1    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    . 0    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 0    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 0    , 1    , 0    ,
     . 1    , 1    , 0    , 1    , 1    , 1    , 1    , 1    ,
     . 0    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 0    , 1    , 1    , 1    , 1    , 1    , 1    /
      data (PenCommand(i),i= 5121, 5200)/
     . 1    , 1    , 0    , 1    , 1    , 1    , 1    , 1    ,
     . 0    , 1    , 1    , 1    , 1    , 1    , 1    , 0    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 0    , 1    . 1    , 1    , 1    ,
     . 0    , 1    , 1    , 1    , 1    , 1    , 0    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 0    , 1    , 1    , 1    ,
     . 1    , 1    , 0    , 1    , 1    , 1    , 1    , 1    .
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 0    ,
     . 1    , 1    , 1      1    , 1    , 1    , 1    , 1    /
      data (PenCommand(i),i= 5201, 5280)/
     . 1    , 1    , 1    , 0    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 0    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 0    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 0    , 1    , 1    . 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 0    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 0    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 0    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 0    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 0    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 0    , 1    /
      data (PenCommand(i),i= 5281, 5360)/
     . 1    , 1    , 1    . 1    , 1    , 1    , 0    , 1    ,
     . 1    . 1    , 1    , 1    , 0    , 1    , 1    , 1    ,
     . 1    . 1    , 1    , 0    , 1    , 1    , 1    , 1    ,
     . 0    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 0    , 1    . 1    , 1    , 1    , 0    , 1    , 1    ,
     . 1    , 1    . 1    , 0    , 1    , 1    , 1    , 1    ,
     . 1    , 0    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 0    , 1    , 1    , 1    , 1    , 1    , 1    , 0    ,
     . 1    , 1    , 1    , 1    , 0    , 1    , 1    , 1    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 0    ,
      data (PenCommand(i),i= 5361, 5440)/
     . 1    , 1    , 0    , 1    , 1    , 1    , 1    , 0    ,
     . 1    , 1    , 1    , 1    , 0    , 1    , 1    , 1    ,
     . 1    , 0    . 1    , 1    . 0    , 1    , 0    , 1    ,
     . 0    . 1    . 0    , 1    , 1    , 0    , 1    , 1    ,
     . 1    . 1    , 1    , 1    . 0    , 1    , 1    , 1    ,
     . 0    . 1    . 1    , 1    , 1    . 1    . 1    . 1    .
     . 0    . 1    [illegible] . 1    . [illegible] . [illegible] . 1    . 1    ,
     . 1    , 0    , 1    , 0    . 1    , 1    , [illegible] . 1    ,
     . 1    . 1    , 1    . 0    , 1    , [illegible] . 1    . 1    .
     . 1    , 1    . 1    . [illegible] , 1    . 1    , 1    , 1    .
      data (PenCommand(i),i= 5441, 5520)/
     . 1    . 1    [illegible] . 1    , 1    . 1    , 1    . 0    .
     . 1    , 1    , 1    , 1    , 1    , 1    , 0    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 0    , 1    , 1    .
     . 1    , 1    . 1    . 1    , 1      0    , 1    , 1    ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i= 5521, 5600)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 0      /
      data (PenCommand(i),i= 5601, 5680)/
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 1      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      /
      data (PenCommand(i),i= 5681, 5760)/
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 0      /
      data (PenCommand(i),i= 5761, 5840)/
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      /
      data (PenCommand(i),i= 5841, 5920)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 1      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      /
      data (PenCommand(i),i= 5921, 6000)/
     . 0      , 1      , 0      , 1      , 0      , 1      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 1      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      /
      data (PenCommand(i),i= 6001, 6080)/
     . 0      , 1      , 1      , 1      , 0      , 1      , 0      , 1      ,
     . 1      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 0      , 1      /
      data (PenCommand(i),i= 6081, 6160)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 0      , 1      ,
     . 1      , 0      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      /
      data (PenCommand(i),i= 6161, 6240)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      /
      data (PenCommand(i),i= 6241, 6320)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i= 6321, 6400)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i= 6401, 6480)/
     . 1    , 0    , 1    , 1    , 1    , 0    , 1    , 1    ,
     . 1    , 1    , 1    , 0    , 1    , 1    , 1    , 1    ,
     . 0    , 1    , 1    , 1    , 1    , 1    , 0    , 1    ,
     . 1    , 1    , 0    , 1    , 1    , 1    , 1    , 1    ,
     . 0    , 1    , 1    , 1    , 1    , 0    , 1    , 1    ,
     . 1    , 0    , 1    , 1    , 1    , 1    , 0    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 0    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 0    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 0    , 1    , 1    , 1    /
      data (PenCommand(i),i= 6481, 6560)/
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 0    , 1    , 0    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 0    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 0    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 0    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 0    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 0    , 1    , 1    , 1    /
      data (PenCommand(i),i= 6561, 6640)/
     . 1    , 1    , 1    , 1    , 0    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 0    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 0    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 0    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 0    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 0    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 0    , 1    , 1    ,
     . 1    , 1    , 1    , 0    , 1    , 1    , 1    , 1    /
      data (PenCommand(i),i= 6641, 6720)/
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 0    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 0    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 0    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 0    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 0    , 1    , 1    , 1    , 1    , 0    ,
     . 1    , 1    , 1    , 0    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 0    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 0    , 1    , 1    , 1    , 1    , 1    /
      data (PenCommand(i),i= 6721, 6800)/
     . 1    , 1    , 1    , 1    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 1    , 1    , 0    ,
     . 1    , 0    , 1    , 1    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 0    /
      data (PenCommand(i),i= 6801, 6880)/
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 1    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 1    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 1    , 1    , 0    ,
     . 1    , 1    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 1    ,
     . 0    , 1    , 0    , 1    , 0    , 1    , 0    , 1    ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       /
      data (PenCommand(i),i= 6881, 6960)/
     . 1       , 1       , 1       , 1       , 1       , 0       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 0       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 0       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 0       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 0       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 0       , 1       , 0       , 1       ,
     . 0       , 1       , 0       , 1       , 0       , 1       , 0       , 1       /
      data (PenCommand(i),i= 6961, 7040)/
     . 0       , 1       , 0       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 0       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 0       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 0       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 0       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 0       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 0       , 1       , 1       , 1       , 1       , 1       ,
     . 0       , 1       , 1       , 1       , 1       , 1       , 0       , 1       /
      data (PenCommand(i),i= 7041, 7120)/
     . 1       , 1       , 1       , 1       , 0       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 0       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 0       , 1       ,
     . 1       , 1       , 1       , 0       , 1       , 0       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 0       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 0       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 0       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 0       , 1       , 1       , 1       , 1       , 1       , 1       , 1       /
      data (PenCommand(i),i= 7121, 7200)/
     . 1       , 0       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 0       , 1       , 1       , 0       , 1       ,
     . 0       , 1       , 1       , 1       , 1       , 0       , 1       , 1       ,
     . 1       , 1       , 0       , 1       , 0       , 1       , 1       , 1       ,
     . 0       , 1       , 0       , 1       , 1       , 1       , 1       , 0       ,
     . 1       , 1       , 1       , 0       , 1       , 1       , 1       , 0       ,
     . 1       , 0       , 1       , 0       , 1       , 1       , 1       , 0       ,
     . 1       , 0       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 0       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 0       , 1       , 1       /
      data (PenCommand(i),i= 7201, 7280)/
     . 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 1       , 0       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 0       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 0       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 1       , 0       ,
     . 1       , 1       , 0       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 0       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1       , 1       , 1
      data (PenCommand(i),i= 7281, 7360)
     . 1       , 1       , 1       , 1       , 1       , 0       , 1       , 1       ,
     . 0       , 1       , 1       , 1       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 0       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 0       , 1       , 1       , 1       , 1       ,
     . 0       , 1       , 1       , 0       , 1       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 0       , 1       , 1       , 1       ,
     . 1       , 1       , 1       , 1       , 1       , 1         0       , 1       ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 1      /
      data  (PenCommand(i),i= 7361, 7440)/
     . 0      , 1      , 0      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 0      , 1      , 1      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 1      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 1      /
      data  (PenCommand(i),i= 7441, 7520)/
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 0      /
      data  (PenCommand(i),i= 7521, 7600)/
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 0      , 1      , 1      , 1      /
      data  (PenCommand(i),i= 7601  7680)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      /
      data  (PenCommand(i),i= 7681, 7760)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      /
      data  (PenCommand(i),i= 7761, 7840)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1     , 0     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 7841, 7920)/
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     . 0     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 7921, 8000)/
     . 0     , 1     . 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 0     /
      data (PenCommand(i),i= 8001, 8080)/
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     . 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 8081, 8160)/
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     . 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     . 1     , 1     /
      data (PenCommand(i),i= 8161, 8240)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     . 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     .
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     .
     . 1     . 1     , 1     , 1     , 1     . 1     , 1     , 1     .
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     .
     . 1     , 1     , 1     , 1     . 1     , 1     . 0     , 1     .
     . 1     , 1     , 1     , 1     , 1     . 1     . 1     , 1     .
     . 1     , 1     , 1     , 1     , 1     . 1     . 1     , 1
      data (PenCommand(i),i= 8241, 8320)/
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     .
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 8321, 8400)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 8401, 8480)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 8481, 8560)/
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 8561, 8640)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     /
      data (PenCommand(i),i= 8641, 8720)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 1     , 0     , 1     , 0     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 8721, 8800)/
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     /
      data (PenCommand(i),i= 8801, 8880)/
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     /
      data (PenCommand(i),i= 8881, 8960)/
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     /
      data (PenCommand(i),i= 8961, 9040)/
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 9041, 9120)/
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     /
      data (PenCommand(i),i= 9121, 9200)/
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0       1     , 0     , 1     , 0     , 1     /
```

```
          data (PenCommand(i),i= 9201, 9280)/
         . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
         . 1      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1
          data (PenCommand(i),i= 9281, 9360)/
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 1      , 1      , 1      , 1      /
          data (PenCommand(i),i= 9361, 9440)/
         . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 1      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 1      , 1      , 0      , 1      , 0      , 1      ,
         . 0      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
         . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
         . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
         . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
         . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      /
          data (PenCommand(i),i= 9441, 9520)/
         . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
         . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
         . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
         . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
         . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
         . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
         . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
         . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
         . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
         . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      /
          data (PenCommand(i),i= 9521, 9600)/
         . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
         . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
         . 1      , 1      , 1      , 0      , 1      , 0      , 1      , 0      ,
         . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
         . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
         . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 1      ,
         . 1      , 0      , 1      , 0      , 1      , 1      , 1      , 1      ,
         . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 0      ,
         . 1      , 0      , 1      , 1      , 1      , 0      , 1      , 0      ,
         . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0
          data (PenCommand(i),i= 9601, 9680)
         . 1      , 1      , 1      , 0      , 1      , 0      , 1      , 0      ,
         . 1      , 0      , 1      , 1      , 1      , 0      , 1      , 0      ,
         . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
         . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
         . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
         . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
         . 1      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
         . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1
      data (PenCommand(i),i= 9681, 9760)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 0     ,
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 0     , 1     , 1
      data (PenCommand(i),i= 9761, 9840)/
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 0     , 1     , 1     , 1     , 0     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i= 9841, 9920)/
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     /
      data (PenCommand(i),i= 9921,10000)/
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 0     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     /
      data (PenCommand(i),i=10001,10080)/
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 1     , 0     , 1     , 0     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 0     , 1
      data (PenCommand(i),i=10081,10160)/
     . 0     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1
      data (PenCommand(i),i=10161,10240)/
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1
      data (PenCommand(i),i=10241,10320)/
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1
      data (PenCommand(i),i=10321,10400)/
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1
      data (PenCommand(i),i=10401,10480)/
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1
      data (PenCommand(i),i=10481,10560)/
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0
      data (PenCommand(i),i=10561,10640)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1
      data (PenCommand(i),i=10641,10720)/
     . 1     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 1     , 0     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 0     , 1     , 1
      data (PenCommand(i),i=10721,10800)/
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 0     , 1     , 1
      data (PenCommand(i),i=10801,10880)/
     . 1     , 1     , 1     , 0     , 1     , 0     , 1     , 0     ,
     . 1     , 0     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1
      data (PenCommand(i),i=10881,10960)/
     . 0     , 1     , 1     , 1     , 0     , 1     , 1     , 1     ,
     . 0     , 1     , 0     , 1     , 1     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 1     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 0     , 1     ,
     . 0     , 1     , 0     , 1     , 0     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1
      data (PenCommand(i),i=10961,11040)/
     . 1     , 1     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 0     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 0     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 1     , 1     , 1     , 1     , 1     , 0     , 1
      data (PenCommand(i),i=11041,11120)/
     . 0     , 1     , 0     , 1     , 1     , 1     , 1     , 1     ,
     . 1     , 0     , 1     , 0     , 1     , 1     , 1     , 1     ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 0      , 1      , 1      , 1      ,
     . 0      , 1      , 0      , 1      , 1      , 1      , 0      , 1      ,
     . 0      , 1      , 1      , 1      , 0      , 1      , 0      , 1      /
      data (PenCommand(i),i=11121,11200)/
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 0      , 1      , 1      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1        0      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1        1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i=11201,11280)/
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 1      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 1      /
      data (PenCommand(i),i=11281,11360)/
     . 1      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 0      , 1      , 1      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 1      /
      data (PenCommand(i),i=11361,11440)/
     . 1      , 0      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 0      , 1      , 0      , 1      , 1      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 1      /
      data (PenCommand(i),i=11441,11520)/
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0      /
      data (PenCommand(i),i=11521,11600)/
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 0    , 1    , 0    , 1    . 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 1    , 1    , 1    ,
     . 1    , 0    , 1    , 1    , 1    , 1    , 1    , 0    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 0    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 0    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 0    , 1    , 1    /
      data (PenCommand(i),i=11601,11680)/
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 0    , 1    , 1    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 1    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 1    , 1    , 0    , 1    , 1    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 0    , 1    , 1    , 1    , 1    , 1    , 1    , 1    /
      data (PenCommand(i),i=11681,11760)/
     . 1    , 1    , 1    , 1    , 1    , 0    , 1    , 0    ,
     . 1    , 1    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    . 0    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 0    , 1    ,
     . 0    , 1    , 0    , 1    , 0    , 1    , 0    , 1    ,
     . 1    , 1    , 0    , 1    , 0    , 1    , 0    , 1    ,
     . 0    , 1    , 0    , 1    , 0    , 1    , 0    , 1    ,
     . 0    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    /
      data (PenCommand(i),i=11761,11840)/
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 0    , 1    , 0    , 1    , 1    ,
     . 1    , 0    , 1    , 1    , 1    , 1    , 1    , 0    ,
     . 1    , 0    , 1    , 1    , 1    , 0    , 1    , 0    ,
     . 1    , 0    , 1    , 1    , 1    , 0    , 1    , 0    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    /
      data (PenCommand(i),i=11841,11920)/
     . 0    , 1    , 0    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 0    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 0    , 1    , 1    ,
     . 1    , 0    , 1    , 0    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 0    ,
     . 1    , 0    . 1    , 0    , 1    , 0    , 1    . 0    ,
     . 1    , 0    . 1    , 0    , 1    . 0    , 1    , 1    /
      data (PenCommand(i).i=11921.12000)/
     . 1    . 1    , 0    . 1    . 0    , 1    , 1    , 1    .
     . 1    . 1    . 1    , 1    . 0    . 1    . 1    . 1    .
     . 1    , 1    . 1    , 1    , 1    . 1    , 0    . 1    .
     . 0    , 1    . 1    . 1    , 1    , 1    , 1    , 1    .
     . 1    , 1    . 0    . 1    . 1    . 1    , 0    , 1    .
     . 0    , 1    , 0    . 1    . 0    , 1    , 0    , 1    .
     . 0    , 1    , 0    , 1    , 1    , 1    . 1    , 1    ,
     . 1    , 1    , 1    , 1    , 1    , 1    , 1    , 1    ,
     . 1    , 1    , 1    , 0    , 1    . C    , 1    , 0    ,
. 1      , 0      , 1      , 1      , 1      , 1      , 1      , 0       /
 data  (PenCommand(i),i=12001,12080)/
. 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0       ,
. 1      , 0      , 1      , 0      , 1      , 1      , 1      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 1      , 1      , 0      , 1      , 1      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 1      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 1      , 1      , 1      , 0      , 1      , 1       ,
. 1      , 1      , 1      , 1      , 0      , 1      , 0      , 1       /
 data  (PenCommand(i),i=12081,12160)/
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 1      , 1      , 1      , 0       ,
. 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0       ,
. 1      , 0      , 1      , 1      , 1      , 0      , 1      , 0       ,
. 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0       ,
. 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0       ,
. 1      , 0      . 1      , 0      , 1      , 0      , 1      , 0       ,
. 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1       ,
. 1      , 1      , 1      , 1      , 1      , 1      , 1      , 0       ,
. 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1       /
 data  (PenCommand(i),i=12161,12240)/
. 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1       ,
. 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1       ,
. 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 1      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 1      , 1      , 1      , 1       ,
. 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1       ,
. 1      , 0      , 1      , 1      , 1      , 1      , 1      , 1       ,
. 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1       /
 data  (PenCommand(i),i=12241,12320)/
. 0      , 1      , 1      , 1      , 1      , 1      , 1      , 1       ,
. 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1       ,
. 1      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       /
 data  (PenCommand(i),i=12321,12400)/
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      . 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      , 1      , 0      , 1      . 0      , 1      , 0      , 1       ,
. 0      , 1      . 0      , 1      . 0      . 1      . 0      . 1       .
. 0      , 1      . 0      . 1      .        . 1      . 0      , 1
 data  (PenCommand(i),i=12401,12480)
. 0      . 1      , 0      . 1      . 0      . 1      . 0      , 1       .
. 0      . 1      , 0      . 1      , 0      . 1      . 0      , 1       .
. 0      . 1      . 0      . 1      , 0      . 1      . 0      , 1       .
. 0      , 1      , 0      . 1      , 0      , 1      . 0      , 1       .
. 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1       ,
. 0      . 1      , 0      , 1      , 0      . 1      , 0      , 1       .
. 0      . 1      , 0      , 1      , 0      . 1      , 0      , 1       .
```

```
     . 0      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      /
      data (PenCommand(i),i=12481,12560)/
     . 0      , 1      , 1      , 1      , 0      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 1      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 0      , 1      ,
     . 1      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 0      , 1      , 0      , 1      , 0      , 1      /
      data (PenCommand(i),i=12561,12640)/
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 0      , 1      ,
     . 0      , 1      , 1      , 1      , 0      , 1      , 0      , 1      ,
     . 1      , 1      , 0      , 1      , 1      , 1      , 1      , 1      /
      data (PenCommand(i),i=12641,12720)/
     . 1      , 1      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 1      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      /
      data (PenCommand(i),i=12721,12800)/
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      /
      data (PenCommand(i),i=12801,12880)/
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      /
      data (PenCommand(i),i=12881,12960)/
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      . 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      /
      data (PenCommand(i),i=12961,13040)/
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      /
      data (PenCommand(i),i=13041,13120)/
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 0      , 1      , 0      , 1      , 0      , 1      , 0      ,
     . 1      , 1      , 1      , 1      , 1      , 1      , 1      , 1      ,
     . 1      , 1      , 1      , 1      , 0      , 1      , 1      , 1      /

      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-------------------------------------------------------------------------------
      subroutine ReadFlightData
c-------------------------------------------------------------------------------
c     read flight (trajectory) data from an input file specified via SFOpen

c.....common block definition files

      include 'MapLim.inc'
      include 'TicDat.inc'
      include 'TrjCom.inc'
      include 'TrjLim.inc'

c.....pointer for QuickDraw globals

      common / QDGPtr /            QDG
      pointer / QDGlobals          QDG
      integer*4                    jQDGlobals
      external                     jQDGlobals

c.....cursor handle

      record / CursHandle /        CursorHndl

c.....I/O status flags

```

```
      integer*2                   ioflag
      integer*2                   iOpen

c.....character items

      character*255               ChrDat
      character*4                 FilTyp
      character*4                 fMaker

c--------------------------------------------------------------------------------

c.....get text data file and open it via routine SFOpenFile.  Exit program if
c     unsuccessful.

      if ( iOpen.eq.0 ) then
         call SFOpenFile ( 8 , iOpen , FilTyp , fMaker )
         if ( iOpen.eq.0 ) then
            call exit
         end if
      end if

c.....use watch cursor while reading data

      cursorHndl = GetCursor ( %val(4) )
      call SetCursor ( %val(cursorHndl.CRHDL^.CRPTR^) )

c.....execute the read loop

      ioflag = 0
      ntrpts = 0
      do while ( ioflag.eq.0 )
         ntrpts = ntrpts + 1
         read(8,*,iostat=ioflag) TofTab(ntrpts),LngTab(ntrpts),LatTab(ntrpts),
     &                           AltTab(ntrpts),JmpTab(ntrpts)
         if ( ioflag.ne.0 ) then
            ntrpts = ntrpts - 1
         end if
      end do

c.....close the input file

      close ( unit=8 )
      iOpen  = 0

c.....determine minimum and maximum limits of data

      if ( ntrpts.ne.0 ) then
         do i = 1 , ntrpts
            if ( i.eq.1 ) then
               MinTof = TofTab (1)
               MinLng = LngTab (1)
               MinLat = LatTab (1)
               MaxTof = TofTab (1)
               MaxLng = LngTab (1)
               MaxLat = LatTab (1)
            else
               MinTof = amin1 ( MinTof , TofTab (i) )
               MinLng = amin1 ( MinLng , LngTab (i) )
               minLat = amin1 ( MinLat , LatTab (i) )
               MaxTof = amax1 ( MaxTof , TofTab (i) )
               MaxLng = amax1 ( MaxLng , LngTab (i) )
               MaxLat = amax1 ( MaxLat , LatTab (i) )
            end if
         end do
      end if

c.....determine preliminary plot limits ( not yet enabled )

      call AutoScale ( MinTof , MaxTof , ndivmj , tMapMn , tMapMx ,
     &                 tDivMj , tDivMi )

c.....reset cursor to arrow

      call SetCursor ( %val(QDG^.Arrow) )

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main
c-----------------------------------------------------------------------
Segment Main
      subroutine RefreshRunDialog
c-----------------------------------------------------------------------

!!SETC USINGINCLUDES = FALSE
      implicit none

c.....common block definition files

      include 'Globals.inc'
      include 'RunSetup.inc'

c.....temporary storage

      record / Rect / refreshTempRect

c-----------------------------------------------------------------------

c.....set to the run dialog port

      call SetPort ( %val( GetSelection ) )

c.....save the current contents of tempRect

      refreshTempRect = tempRect

c.....draw the Close button

      call GetDItem( %val(GetSelection), %val(rCloseButton),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )

c.....draw thick default outline

      call PenSize( %val(3), %val(3) )

c.....draw outside the button by one pixel

```

```
      call InsetRect( %ref(tempRect), %val(-4), %val(-4) )

c.....draw the button outline

      call FrameRoundRect( %ref(tempRect), %val(16), %val(16) )

c.....restore the pen size to the default value

      call Pensize( %val(1), %val(1) )

c.....draw a line

      call MoveTo ( %val(82), %val(65)  )
      call LineTo ( %val(82), %val(104) )

c.....restore tempRect

      tempRect = refreshTempRect

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c--------------------------------------------------------------------------------
      subroutine ResizeTheMap
c--------------------------------------------------------------------------------
c     resize the map window

c.....include common block definition files

      include 'FntCom.inc'
      include 'MapCom.inc'

c.....item stuff

      record / handle /       ItHndl
      record / rect /         ItRect
      integer*4               ItType
      integer*2               ItNmbr
      string*255              ItText

c....."resize the map" dialog interface records

      common / ResizeMap /    ResizeMapPtr,    iGotResizePtr
      record / DialogPtr /    ResizeMapPtr
      integer*2               iGotResizePtr

c.....character strings

      character*255           ChrDat

c.....dialog interface variables ( note that pointers are i*4 )

      integer*4               infront

c.....dialog interface values
```

```

      data     infront / -1 /

c.....set dialog font to Times ( it is the most compact )

      FntNam = 'Times'
      call GetFNum ( %val(FntNam) , FntNum )
      call setDAfont ( %val(FntNum) )

c.....Get resize map dialog

      if ( iGotResizePtr.eq.0 ) then
         ResizeMapPtr   = GetNewDialog ( %val(135) , %val(nil) , %val(inFront) )
         iGotResizePtr = 1
      end if
      call SetPort ( %val(ResizeMapPtr) )

c.....bring the dialog window to the front

      call ShowWindow ( %val(ResizeMapPtr) )
      call SelectWindow ( %val(ResizeMapPtr) )

c.....Highlight the OK button

      ItNmbr = 1
      call GetDItem ( %val(ResizeMapPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call PenSize ( %val(3) , %val(3) )
      call InsetRect ( %ref(ItRect) , %val(-4) , %val(-4) )
      call FrameRoundRect ( %ref(ItRect) , %val(18) , %val(18) )

c.....set and select map window width

      ItNmbr = 3
      write(ChrDat,*) MapWidth
      ItText = ChrDat
      call GetDItem ( %val(ResizeMapPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call SetIText ( %val(ItHndl) , %val(ItText) )
      call SelIText ( %val(ResizeMapPtr) , %val(ItNmbr) , %val(0) , %val(32767) )

c.....set map window height

      ItNmbr = 4
      write(ChrDat,*) MapHeight
      ItText = ChrDat
      call GetDItem ( %val(ResizeMapPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call SetIText ( %val(ItHndl) , %val(ItText) )

c.....loop until either the OK button or the RESET button is clicked.
c     Monitor all other relevant events and update the dialog as necessary.

      ItNmbr = 0
      do while ( ItNmbr.ne.1 .and. ItNmbr.ne.2 )

c.......get number of item hit

         call ModalDialog ( %val(nil) , ItNmbr )

c.......reset to screen limits

         if ( ItNmbr.eq.2 ) then

```

```
c...........width

            ItNmbr = 3
            write(ChrDat,*) DefWidth
            ItText = ChrDat
            call GetDItem ( %val(ResizeMapPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                      %ref(ItHndl) , %ref(ItRect) )
            call SetIText ( %val(ItHndl) , %val(ItText) )
            call SelIText ( %val(ResizeMapPtr) , %val(ItNmbr) , %val(0) ,
     .                      %val(32767) )

c...........height

            ItNmbr = 4
            write(ChrDat,*) DefHeight
            ItText = ChrDat
            call GetDItem ( %val(ResizeMapPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                      %ref(ItHndl) , %ref(ItRect) )
            call SetIText ( %val(ItHndl) , %val(ItText) )

         end if

      end do

c.....get map window width

      ItNmbr = 3
      call GetDitem ( %val(ResizeMapPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call GetIText ( %val(ItHndl) , %val(ItText) )
      ChrDat = ItText
      if ( ChrDat.ne.' ' ) then
         read(ChrDat,*,iostat=ioflag) tmpl
         if ( ioflag.ne.0 ) then
            tmpl   = 0.0
         end if
      else
         tmpl   = 0.0
      end if
      MapWidth = tmpl

c.....get map window height

      ItNmbr = 4
      call GetDitem ( %val(ResizeMapPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call GetIText ( %val(ItHndl) , %val(ItText) )
      ChrDat = ItText
      if ( ChrDat.ne.' ' ) then
         read(ChrDat,*,iostat=ioflag) tmpl
         if ( ioflag.ne.0 ) then
            tmpl   = 0.0
         end if
      else
         tmpl   = 0.0
      end if
      MapHeight = tmpl

c.....hide dialog

      call HideWindow ( *val(ResizeMapPtr) )

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-----------------------------------------------------------------------
      subroutine RunSetupDialog
c-----------------------------------------------------------------------

!!SETC USINGINCLUDES = FALSE
      implicit none

c.....common block definition files

      include 'FileInfo.inc'
      include 'FileMenu.inc'
      include 'Globals.inc'
      include 'MapMenu.inc'
      include 'RunSetup.inc'
      include 'Traj.inc'
      include 'TrjCom.inc'

c.....set up pointer for QuickDraw globals

      pointer / QDGlobals /      qdg
      common  / QDGPtr    /      qdg

c.....external function declarations

      external  RunSetupFilter
      logical*1 RunSetupFilter

      external  CollectRunInput
      integer*2 CollectRunInput

c.....integers for error handling

      integer*2 ErrorItem
      integer*2 iopen
      integer*4 ioserr

c.....integer for popup menu selection

      integer*2 theSelection

c.....boolean for dialog handle loop

      logical*1 ExitDialog

c.....intermediate text strings

      string*255 ItemText
      string*255 Filename

c.....character intermediate

      character*255 CharData

c.....other
```

```

      character*4 FilTyp
      character*4 fmaker

c.....cursor handle

      record / CursHandle /       CursorHndl

c.....STR handling

      record / handle       /     RezHndl
      record / StringHandle /     STRHndl

      string*255              tempString

c.....pop-up menu handling

      record / MenuHandle /   PopupHndl

c.....dialog template handling

      record / DialogTHndl    / DlogHndl
      record / DialogTemplate / DlogTemp
      record / Rect           / RunRect

c.....screen position info

      integer*2                 menuHeight
      integer*4                 left, bottom, top, right
      integer*4                 dialogHeight, dialogWidth

c.....dimensions of runsetup dialog box

      parameter           ( dialogHeight        =  302 )
      parameter           ( dialogWidth         =  491 )

c-------------------------------------------------------------------------------

c.....get the previous grafPort

      call GetPort ( %val( SavedPort ))

c.....get the handle and the template for the dialog resource

      RezHndl  = GetResource( %val('DLOG'), %val(rRunSetupDLOG) )
      call HNoPurge( %val(RezHndl) )
      DlogHndl = RezHndl
      call HNoPurge( %val(DlogHndl) )

c.....get the menuHeight (don't assume it is fixed at 20!)

      menuHeight = GetMBarHeight()

c.....get the screen extents (use i*4 for screen math per Mac Tech Note 117)

      left   = qdg^.screenBits.bounds.left
      right  = qdg^.screenBits.bounds.right
      bottom = qdg^.screenBits.bounds.bottom
      top    = qdg^.screenBits.bounds.top + int4(menuHeight)

c.....set the four corners of the dialog

      RunRect.left   = int2(  ( (right - left) - dialogWidth  ) / 2 )
      RunRect.top    = int2(( ( (bottom - top) - dialogHeight ) / 2 ) + menuHeight)
      RunRect.bottom = RunRect.top  + int2(dialogHeight)
      RunRect.right  = RunRect.left + int2(dialogWidth)

      DlogHndl.DtH^.DtP^.boundsRect = RunRect
      call ChangedResource( %val(RezHndl) )
      call WriteResource( %val(RezHndl) )
      call HPurge( %val(RezHndl) )
      call HPurge( %val(DlogHndl) )

c.....bring in the dialog resources

      GetSelection = GetNewDialog( %val(rRunSetupDLOG),
     &                            %val(nil), %val(inFront) )
      call ShowWindow   ( %val(GetSelection) )
        call SelectWindow( %val(GetSelection) )
      call SetPort      ( %val(GetSelection) )

        TheDialogPtr.DPk = GetSelection.DP
      ThisEditText.TEH = TheDialogPtr.DPk
      call HLock( %val(ThisEditText))

c.....set the TE point size
      ThisEditText.TEH^.TEP^.txSize = int2(12)

c.....set the window point size with QuickDraw procedure
      call TextSize( %val(int2(12)) )

c.....set the TE font ID
      ThisEditText.TEH^.TEP^.txFont = systemFont

c.....set the window font ID with QuickDraw procedure
      call TextFont( %val(systemFont) )

c.....set the TE font ascent, descent and leading values
      ThisEditText.TEH^.TEP^.fontAscent = int2(12)
      ThisEditText.TEH^.TEP^.lineHeight = int2(12 + 3 + 1)

      call HUnlock( %val(ThisEditText) )

c.....set up initial conditions for dialog items

      rDegreeSelection   = rDegWestButton
      rDistanceSelection = rMeters
      rTimeSelection     = rSeconds
      AscentSelection    = 'Ascent Profile'
      ClimateSelection   = 1

c.....set up degrees-west button

      call GetDItem( %val(GetSelection), %val(rDegWestButton),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )
      CItem.CtlH = DItem.bhdl
      call SetCtlValue( %val(CItem), %val(int2(1) ) )
      call SetupTheItem( GetSelection, rDegWestButton,
     &                   .true., .true., .true., .false., tempRect, 0, 0 )

c.....set up degrees-east button

      call GetDItem( %val(GetSelection), %val(rDegEastButton),
     &               %ref(DType), %ref(DItem), %ref(tempRect) )
      CItem.CtlH = DItem.bhdl
      call SetCtlValue( %val(CItem), %val(int2(0) ) )
      call SetupTheItem( GetSelection, rDegEastButton,
     &                   .true., .true., .true., .false., tempRect, 0, 0 )
```

```

c.....set up meters button

       call GetDItem( %val(GetSelection), %val(rMeters),
     &                %ref(DType), %ref(DItem), %ref(tempRect) )
       CItem.CtlH = DItem.bhdl
       call SetCtlValue( %val(CItem), %val( int2(1) ) )
       call SetupTheItem( GetSelection, rMeters,
     &                    .true., .true., .true., .false., tempRect, 0, 0 )

c.....set up kilometers button

       call GetDItem( %val(GetSelection), %val(rKilometers),
     &                %ref(DType), %ref(DItem), %ref(tempRect) )
       CItem.CtlH = DItem.bhdl
       call SetCtlValue( %val(CItem), %val( int2(0) ) )
       call SetupTheItem( GetSelection, rKilometers,
     &                    .true., .true., .true., .false., tempRect, 0, 0 )

c.....set up seconds button

       call GetDItem( %val(GetSelection), %val(rSeconds),
     &                %ref(DType), %ref(DItem), %ref(tempRect) )
       CItem.CtlH = DItem.bhdl
       call SetCtlValue( %val(CItem), %val(int2(1) ) )
       call SetupTheItem( GetSelection, rSeconds,
     &                    .true., .true., .true., .false., tempRect, 0, 0 )

c.....set up minutes button

       call GetDItem( %val(GetSelection), %val(rMinutes),
     &                %ref(DType), %ref(DItem), %ref(tempRect) )
       CItem.CtlH = DItem.bhdl
       call SetCtlValue( %val(CItem), %val(int2(0) ) )
       call SetupTheItem( GetSelection, rMinutes,
     &                    .true., .true., .true., .false., tempRect, 0, 0 )

c.....set up hours button

       call GetDItem( %val(GetSelection), %val(rHours),
     &                %ref(DType), %ref(DItem), %ref(tempRect) )
       CItem.CtlH = DItem.bhdl
       call SetCtlValue( %val(CItem), %val(int2(0) ) )
       call SetupTheItem( GetSelection, rHours,
     &                    .true., .true., .true., .false., tempRect, 0, 0 )

c.....set up wind model selector

       call GetDItem( %val(GetSelection), %val(rWindModelSelector),
     &                %ref(DType), %ref(DItem), %ref(tempRect) )
       CItem.CtlH = DItem.bhdl
       call SetCtlMax( %val(CItem), %val(rWindModelSelectorPopup) )
       call SetCtlValue( %val(CItem), %val(ClimateSelection) )
       call SetupTheItem( GetSelection, rWindModelSelector,
     &                    .true., .true., .true., .false., tempRect, 0, 0 )

c.....set up ascent selector

       call GetDItem( %val(GetSelection), %val(rAscentSelector),
     &                %ref(DType), %ref(DItem), %ref(tempRect) )
       CItem.CtlH = DItem.bhdl
       call SetCtlMax( %val(CItem), %val(rAscentSelectorPopup) )
       call SetCtlValue( %val(CItem), %val(int2(1) ) )
       call SetupTheItem( GetSelection, rAscentSelector,
     &                      .true., .true., .true., .false., tempRect, 0, 0 )

c.....check to see if an old file's data is available

      if ( iGotOldFile ) then

c.......get the old value of latitude

       call GetDItem( %val(GetSelection), %val(rLatitude),
     &                    %ref(DType), %ref(DItem), %ref(tempRect) )
       RezHndl = GetResource( %val('STR '), %val(rOldLatitude) )
       StrHndl.shdl = RezHndl.bhdl
       tempString = StrHndl.shdl^.sptr^
       call SetIText( %val(DItem), %val(tempString) )

c.......get the old value of longitude

       call GetDItem( %val(GetSelection), %val(rLongitude),
     &                    %ref(DType), %ref(DItem), %ref(tempRect) )
       RezHndl = GetResource( %val('STR '), %val(rOldLongitude) )
       StrHndl.shdl = RezHndl.bhdl
       tempString = StrHndl.shdl^.sptr^
       call SetIText( %val(DItem), %val(tempString) )

c.......get the old value of flight duration

       call GetDItem( %val(GetSelection), %val(rDuration),
     &                    %ref(DType), %ref(DItem), %ref(tempRect) )
       RezHndl = GetResource( %val('STR '), %val(rOldDuration) )
       StrHndl.shdl = RezHndl.bhdl
       tempString = StrHndl.shdl^.sptr^
       call SetIText( %val(DItem), %val(tempString) )

c.......get the old value of altitude

       call GetDItem( %val(GetSelection), %val(rAltitude),
     &                    %ref(DType), %ref(DItem), %ref(tempRect) )
       RezHndl = GetResource( %val('STR '), %val(rOldAltitude) )
       StrHndl.shdl = RezHndl.bhdl
       tempString = StrHndl.shdl^.sptr^
       call SetIText( %val(DItem), %val(tempString) )

c.......get the old value of Mission text

       call GetDItem( %val(GetSelection), %val(rMissionLabel),
     &                    %ref(DType), %ref(DItem), %ref(tempRect) )
       RezHndl = GetResource( %val('STR '), %val(rOldMissionText) )
       StrHndl.shdl = RezHndl.bhdl
       tempString = StrHndl.shdl^.sptr^
       call SetIText( %val(DItem), %val(tempString) )

c.......get the old value of wind model selection

       call GetDItem( %val(GetSelection), %val(rWindModelSelector),
     &                    %ref(DType), %ref(DItem), %ref(tempRect) )
       CItem.CtlH = DItem.bhdl
       StrHndl    = GetString( %val(rOldClimate) )
       tempString = StrHndl.shdl^.sptr^
       CharData   = tempString
       read(CharData,*) ClimateSelection
       call SetCtlValue( %val(CItem), %val(ClimateSelection) )

c.......get the old value of ascent profile selection

```

```
        StrHndl         = GetString( %val(rOldAscent) )
        tempString      = StrHndl.shdl^.sptr^
        CharData        = tempString
        PopupHndl.menuH = GetMHandle( %val(rAscentSelectorPopup) )
        call SetItem( %val(PopupHndl), %val(int2(1)), %val(tempString) )

c.......get the old value of the deg West/deg East radio button

        call ClearDegreeGroup
        StrHndl    = GetString ( %val(rOldDegRadio) )
        tempString = StrHndl.shdl^.sptr^
        CharData   = tempString
        read(CharData,*) rDegreeSelection
        call GetDItem( %val(GetSelection), %val(rDegreeSelection),
     &                 %ref(DType), %ref(DItem), %ref(tempRect) )
        CItem.CtlH = DItem.bhdl
        call SetCtlValue( %val(CItem), %val(int2(1) ) )

c.......get the old value of the m/km radio button

        call ClearDistanceGroup
        StrHndl    = GetString ( %val(rOldDistRadio) )
        tempString = StrHndl.shdl^.sptr^
        CharData   = tempString
        read(CharData,*) rDistanceSelection
        call GetDItem( %val(GetSelection), %val(rDistanceSelection),
     &                 %ref(DType), %ref(DItem), %ref(tempRect) )
        CItem.CtlH = DItem.bhdl
        call SetCtlValue( %val(CItem), %val(int2(1) ) )

c.......get the old value of the sec/min/hrs radio button

        call ClearTimeGroup
        StrHndl    = GetString ( %val(rOldTimeRadio) )
        tempString = StrHndl.shdl^.sptr^
        CharData   = tempString
        read(CharData,*) rTimeSelection
        call GetDItem( %val(GetSelection), %val(rTimeSelection),
     &                 %ref(DType), %ref(DItem), %ref(tempRect) )
        CItem.CtlH = DItem.bhdl
        call SetCtlValue( %val(CItem), %val(int2(1) ) )

c.......close the resource file for now

        call CloseResFile( %val(RefNum) )

      endif

c.....call routine to draw any lists, lines, or rectangles

      call RefreshRunDialog

c.....do not exit the dialog handle loop yet

      ExitDialog = .false.

c.....*********************************
c.....** start of dialog handle loop **
c.....*********************************

      do while (.not.ExitDialog)

c.......get number of item hit

```

```
        call ModalDialog ( RunSetupFilter , ItemHit )

c.......check for update

        if( ItemHit .eq. 32000 ) then
          call RefreshRunDialog
          call EndUpdate( %val(GetSelection) )
        else
          call GetDItem( %val(GetSelection), %val(ItemHit),
     &                   %ref(DType), %ref(DItem), %ref(tempRect) )
          CItem.CtlH = DItem.bhdl
        endif

c.......check for Close button

        if( ItemHit .eq. rCloseButton ) then
c.........first enable the New and Open Mission buttons
          call MenuSet( FileMenuID, FileItemNewMission,  .true. )
          call MenuSet( FileMenuID, FileItemOpenMission, .true. )
          ExitDialog = .true.
        endif

c.......check for Run button

        if( ItemHit .eq. rRunButton ) then
          ErrorItem = CollectRunInput()
          if ( ErrorItem.ne.0 ) then
            call SysBeep( %val(int2(20) ) )
            call SeliText( %val(GetSelection), %val(ErrorItem),
     &                     %val(int2(0)), %val(int2(32767)) )
          else
c...........use watch cursor while running
            cursorHndl = GetCursor ( %val(int2(4)) )
            call SetCursor ( %val(cursorHndl.CRHDL^.CRPTR^) )
            call RunTraj( AscentSelection, int4(ClimateSelection),
     &                    xDuration, xLatitude, xLongitude, xAltitude)
c...........reset cursor to arrow
            call SetCursor ( %val(QDG^.Arrow) )
            call SysBeep( %val(int2(20) ) )
            call SysBeep( %val(int2(20) ) )

          endif
        endif

c.......check for Map button

        if( ItemHit .eq. rMapButton ) then
c.........first enable the map menu items
          call MenuSet( MapMenuID, itemGetNewDataSet, .true. )
          call MenuSet( MapMenuID, itemNewMap,        .true. )
          call MenuSet( MapMenuID, itemSaveMap,       .true. )
          call MenuSet( MapMenuID, itemDone,          .true. )

          call MapIt

c.........now disable the entire map menu
          call MenuSet( MapMenuID, 0, .false. )

        endif

c.......check for Save button

        if( ItemHit .eq. rSaveButton ) then
          ErrorItem = CollectRunInput()
          if ( ErrorItem.ne.0 ) then
            call SysBeep( %val(int2(20) ) )
            call SeliText( %val(GetSelection), %val(ErrorItem),
     &                     %val(0), %val(32767) )
          else
            call SaveMissionFile
          endif
        endif

c.......check for degrees-west radio button

        if( ItemHit .eq. rDegWestButton ) then
          call ClearDegreeGroup
          call SetCtlValue( %val(CItem), %val(int2(1) ) )
          rDegreeSelection = rDegWestButton
        endif

c.......check for degrees-east radio button

        if( ItemHit .eq. rDegEastButton ) then
          call ClearDegreeGroup
          call SetCtlValue( %val(CItem), %val(int2(1) ) )
          rDegreeSelection = rDegEastButton
        endif

c.......check for meters radio button

        if( ItemHit .eq. rMeters ) then
          call ClearDistanceGroup
          call SetCtlValue( %val(CItem), %val(int2(1) ) )
          rDistanceSelection = rMeters
        endif

c.......check for kilometers radio button

        if( ItemHit .eq. rKilometers ) then
          call ClearDistanceGroup
          call SetCtlValue( %val(CItem), %val(int2(1) ) )
          rDistanceSelection = rKilometers
        endif

c.......check for seconds radio button

        if( ItemHit .eq. rSeconds ) then
          call ClearTimeGroup
          call SetCtlValue( %val(CItem), %val(int2(1) ) )
          rTimeSelection = rSeconds
        endif

c.......check for minutes radio button

        if( ItemHit .eq. rMinutes ) then
          call ClearTimeGroup
          call SetCtlValue( %val(CItem), %val(int2(1) ) )
          rTimeSelection = rMinutes
        endif

c.......check for hours radio button

        if( ItemHit .eq. rHours ) then
          call ClearTimeGroup
          call SetCtlValue( %val(CItem), %val(int2(1) ) )
          rTimeSelection = rHours
        endif
```

```

c.......check for wind model selection

        if( ItemHit .eq. rWindModelSelector ) then
          theSelection = GetCtlValue( %val(CItem) )
          select case(theSelection)
            case(1)
              ClimateSelection = 1
            case(2)
              ClimateSelection = 2
            case(3)
              ClimateSelection = 3
            case(4)
              ClimateSelection = 4
            case(5)
              ClimateSelection = 5
            case(6)
              ClimateSelection = 6
            case(7)
              ClimateSelection = 7
            case(8)
              ClimateSelection = 8
            case(9)
              ClimateSelection = 9
            case(10)
              ClimateSelection = 10
            case(11)
              ClimateSelection = 11
            case(12)
              ClimateSelection = 12
            case(13)
              ClimateSelection = 13
            case default
              ClimateSelection = 1
          end select
        endif

c.......check for ascent profile selection

        if( ItemHit .eq. rAscentSelector ) then
          theSelection = GetCtlValue( %val(CItem) )

          select case(theSelection)

            case(1)
c.............find what the menu shows and use that name
              PopupHndl.menuH = GetMHandle( %val(rAscentSelectorPopup) )
              call GetItem( %val(PopupHndl), %val(int2(1)), %ref(tempString))
              AscentSelection = tempString

            case(2)
c.............get the name of the user's file; stick it in the popup menu
              call SFOpenAscentFile( iopen, Filename )
              if( iopen.eq.1 ) then
c...............change the menu item
                PopupHndl.menuH = GetMHandle( %val(rAscentSelectorPopup) )
                call SetItem( %val(PopupHndl), %val(int2(1)), %val(Filename) )
                AscentSelection = Filename
              endif
          end select
        endif

c.....*******************************
c.....** end of dialog handle loop **
```

```
c.....*******************************

      end do

c.....restore the port

      call SetPort( %val(SavedPort) )

      call DisposDialog( %val(GetSelection) )

      return
      end


c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main
c-------------------------------------------------------------------------------
Segment Main
      logical*1 function RunSetupFilter ( %val(theDialog), theEvent, ItemHit )
c-------------------------------------------------------------------------------
c     process the relevant events which may occur while the Run Setup dialog
c     is in use.

!!SETC USINGINCLUDES = FALSE
      implicit none

c.....common block definition files

      include 'RunSetup.inc'

c.....other declarations

      record / DialogPtr   / theDialog
      record / EventRecord / theEvent
      record / WindowPtr   / WinPtr
      record / Point       / MyPt

c.....integers

      integer*2 WindowPart
      integer*2 ChCode
      integer*2 CmdCode

c.....character for key handling

      character*1 Ch

c-------------------------------------------------------------------------------

c.....default: Modal Dialog should process the event

      RunSetupFilter = .false.

c.....handle an update event

```

```
      if (theEvent.what .eq. updateEvt) then
        WindowPart = FindWindow ( %val(TheEvent.where) , %ref(WinPtr) )

c.......if in the dialog then only do an update

        if ( WinPtr.WP .eq. theDialog.DP ) then
          call BeginUpdate( %val(theDialog) )
          call DrawDialog(  %val(theDialog) )
          RunSetupFilter = .true.
          ItemHit = 32000
        endif

c.....handle a mouse event

      else if ( theEvent.what .eq. mouseDown ) then
        MyPt = theEvent.where
        call GlobalToLocal ( %ref(MyPt) )

c.....handle a keydown event

      else if ( theEvent.what .eq. keyDown ) then
        ChCode = jiand ( TheEvent.message , CharCodeMask )
        Ch     = char ( ChCode )

c.......check for the Cmd key depression
        CmdCode = jiand ( theEvent.modifiers , CmdKey )

        if ( CmdCode.ne.0 ) then
          if ( Ch.eq.'x' .or. Ch.eq.'X' ) then
            call DlgCut( %val(theDialog) )
            RunSetupFilter = .true.
          else if ( Ch.eq.'c' .or. Ch.eq.'C' ) then
            call DlgCopy( %val(theDialog) )
            RunSetupFilter = .true.
          else if ( Ch.eq.'v' .or. Ch.eq.'V' ) then
            call DlgPaste( %val(theDialog) )
            RunSetupFilter = .true.
          endif
        endif

c.......return ItemHit=1 for <Return> or <Enter>
        if ( ChCode.eq.3 .or. ChCode.eq.13 ) then
          ItemHit        = 1
          RunSetupFilter = .true.
        end if

      endif

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G ToolBox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

      SUBROUTINE RUNTRAJ (INFILE,CLIMATE,FLIGHT_TIME,INIT_LAT,
     &                      INIT_LON, INITIAL_ALT)
C
      EXTERNAL FUNCNT
```

```
C
      LOGICAL FIRST, OFF_EAST, OFF_WEST
      DIMENSION PARMIN(10),XMEAN(10),STDDEV(10)
      DIMENSION CURR_POSITION(3),INIT_POSITION(3)
C
      REAL*4 SLOPE,Y_INTERCEPT,TOTAL_SEG_DIST,SEGMENT_DIST
      REAL*4 INIT_SEG_VEL,FIN_SEG_VEL,FUNCNT,PERM_INT_STEP
      REAL*4 ANGLE_RESOLUTION,DIFFTM,INIT_SEG_TIME,DIFFGT
      REAL*4 FLIGHT_TIME,INITIAL_ALT,INTEGRAT_STEP
      REAL*4 FIN_SEG_TIME,FIN_INTEGRAT_LMT
      REAL*4 DIFLAT,DIFLON,TOLERANCE,INIT_INTEGRAT_LMT
      REAL*4 WIND_AZIMUTH,WIND_VEL,INIT_POSITION,CURR_POSITION
      REAL*4 AE,BE,RADIUS_ERTH,T2TOF.DELTAT,ERTH_ROTATION,R2D,DIFFNC
      REAL*4 XLATSNG,XLNGSNG,GRD_RANGE,UGH,VGH
      REAL*4 LON_ERTH_ROT,INIT_EAST_WIND,INIT_NORTH_WIND,TOTAL_EAST
      REAL*4 DIST_EAST,DIST_NORTH,FINAL_ALT,TOTAL_NORTH
      REAL*4 NEW_LAT,NEW_LON,INIT_LAT,INIT_LON,LAT_DEG,LON_DEG
      REAL*8 FSIM
C
      INTEGER*4 ICHECK,IS,CLIMATE
      INTEGER*4 PERCNT_DONE
C
      LOGICAL*1 SegmentsFinished
C
      CHARACTER INFILE*255, TAB*1
C
      DATA TAB /9/
      DATA R2D /57.29578/
      DATA ERTH_ROTATION  /7.2921151E-05/
      DATA IS /0/

      COMMON /DRVOUT/ FSIM(13),DFSIM(13)
      COMMON /DRVOT2/ ALP   ,ALPDOT,BTA   ,BTADOT,HSIM  ,HDOT  ,
     .                V     ,VDOT  ,X     ,XDOT  ,Y     ,YDOT  ,VI
      EQUIVALENCE (FSIM( 1),T     ),(FSIM(12),LAT_RAD)
      EQUIVALENCE (FSIM(13),LON_RAD)
C
      COMMON /PLTOUT/ HWLS  ,HWLL  ,VDS   ,VTS   ,VUS   ,VDL   ,VTL   ,
     &                VUL   ,RDS   ,RTS   ,RVS   ,RDL   ,RTL   ,RVL   ,
     &                RNDL  ,RNDS  ,RNTL  ,RNTS  ,RNUL  ,RNUS  ,RNVL  ,
     &                RNVS  ,RNDLM ,RNDSM ,RNTLM ,RNTSM ,RNULM ,RNUSM ,
     &                RNVLM ,RNVSM
C
C.... GRAM PROGRAM COMMON BLOCKS.
C
      INCLUDE 'naspgcom.inc'
C
C.... DRYDEN SIM COMMON BLOCKS.
C
      INCLUDE 'naspcom.inc'
C
      include 'traj.inc'
      include 'trjcom.inc'
      include 'globals.inc'

      record / GrafPort    /  SavedPort
      record / rect /         ItRect
      integer*4               ItType
      string*255              ItText
      string*255              PrText
      character*254           Char_Data
      record / DialogPtr /    StatPtr
      record / handle /       ItHndl
      integer*4               last_value
```

```
C
C-----------------------------------------------------------------------
C
      RADIUS_ERTH = 6375416.785
C
      CALL OPENIT(INIT_LAT,INIT_LON,INFILE,FLIGHT_TIME,CLIMATE)
C
      last_value = 0

c.....get the previous grafPort

      call GetPort ( %val( SavedPort ))

c.....display the status dialog

      StatPtr = GetNewDialog ( %val(int2(129)) , %val(nil) , %val(inFront) )
      call ShowWindow  ( %val(StatPtr) )
        call SelectWindow( %val(StatPtr) )
      call SetPort     ( %val(StatPtr) )

      call DrawDialog( %val(StatPtr) )

c.....get the handle for the percent complete text box (item #3)

      PrText = '0 %'
      call GetDItem ( %val(StatPtr) , %val(int2(3)) , %ref(ItType) , %ref(ItHndl) ,
     .                %ref(ItRect) )
      call SetIText ( %val(ItHndl) , %val(PrText) )
c
C.... PERFORM INITIALIZATION.
C
      FIRST    = .TRUE.
      OFF_EAST = .FALSE.
      OFF_WEST = .FALSE.
      TOTAL_SEG_DIST   = 0.0
      FIN_INTEGRAT_LMT = 0.0
      TOTAL_EAST  = 0.0
      TOTAL_NORTH = 0.0
      DELTAT = 0.0
      DIFFGT = 0.0
      DIFFNC = 0.0
      DIFFTM = 0.0
C
C     SUBROUTINE GRAMIN PROVIDES THE BALLOON DRIFT SIMULATION WITH
C     INITIAL VALUES FOR THE LATITUDE (INIT_LAT (DEG)), LONGITUDE
C     (INIT_LON (DEG)), ALTITUDE (H1 (KM)), AND THE RADIUS OF THE EARTH
C     (RADIUS_ERTH (M)).
C
      IF(CLIMATE .NE. 13)THEN
        CALL GRAMIN (INIT_LAT, INIT_LON, INITIAL_ALT, CLIMATE)
      ENDIF
C
C     INITIAL INPUTS FOR RTGRAM AND THE SUBROUTINES.
C
      LAT_RAD = INIT_LAT/R2D
      LON_RAD = INIT_LON/R2D
      XLATSNG = SNGL(LAT_RAD)
      XLNGSNG = SNGL(LON_RAD)
      INIT_INTEGRAT_LMT = 0.0
      TOTAL_EAST  = 0.0
      TOTAL_NORTH = 0.0
      TOLERANCE   = 0.0001
      ANGLE_RESOLUTION = 1.0
      NO_OF_PTS = 1
      AE  = 6378134.999
      BE  = 6356750.499
C
C     NOGAPS/GRAMRT DERIVES THE WIND COMPONENTS, EAST (UGH (M/S))
C     AND NORTH (VGH (M/S)).  THESE COMPONENTS ARE BASED UPON THE
C     INITIAL LATITUDE (INIT_LAT (DEG)/LAT_RAD (RAD)), LONGITUDE
C     (INIT_LON (DEG)/LON_RAD (RAD)), AND ALTITUDE (INTALT
C     (M)/HSIM (FEET)).
C
      IF(CLIMATE .EQ. 13)THEN
        CALL NOGAPS(INITIAL_ALT, INIT_LAT, INIT_LON, UGH, VGH, FIRST)
      ELSE
        HSIM = INITIAL_ALT * 3.28084
        CALL GRAMRT (FIRST)
      ENDIF
C
      IF (VGH.EQ.0.0) THEN
        WIND_AZIMUTH = 90.0
      ELSE
        WIND_AZIMUTH = (ATAN(UGH/VGH))*R2D
      ENDIF
      IF( VGH.LT.0.0) THEN
        WIND_AZIMUTH = 180.0+WIND_AZIMUTH
      ELSEIF( UGH.LT.0.0 ) THEN
        WIND_AZIMUTH = 360.0+WIND_AZIMUTH
      ENDIF
C
      WIND_VEL = SQRT((UGH**2) + (VGH**2))
C
      INIT_EAST_WIND = UGH
      INIT_NORTH_WIND = VGH
C
      INIT_POSITION(1)  = ((RADIUS_ERTH)*COS(XLATSNG))*SIN(XLNGSNG)
      INIT_POSITION(2)  = ((RADIUS_ERTH)*COS(XLATSNG))*COS(XLNGSNG)
      INIT_POSITION(3)  =  (RADIUS_ERTH)*SIN(XLATSNG)
C
      CURR_POSITION(1) = INIT_POSITION(1)
      CURR_POSITION(2) = INIT_POSITION(2)
      CURR_POSITION(3) = INIT_POSITION(3)
C
      CALL GRNRGE(INIT_POSITION,CURR_POSITION,AE,BE,RADIUS_ERTH,
     &               2,2,GRD_RANGE)
C
C     PLACE VALUES IN COMMON BLOCK FOR WRITING TO EXTERNAL PLOT FILE
C
      TIME_ARRAY(1)     = INIT_INTEGRAT_LMT
      LAT_ARRAY(1)      = INIT_LAT
      LON_ARRAY(1)      = INIT_LON
      ALT_ARRAY(1)      = INITIAL_ALT
      GRANGE_ARRAY(1)   = GRD_RANGE
      WINDAZ_ARRAY(1)   = WIND_AZIMUTH
      WIND_VEL_ARRAY(1) = WIND_VEL
C
C     PLACE VALUES IN COMMON BLOCK FOR MAKING MAP
C
      TOFTAB(1) = INIT_INTEGRAT_LMT
      LNGTAB(1) = -1.0 * INIT_LON
      LATTAB(1) = INIT_LAT
      ALTTAB(1) = INITIAL_ALT
      JMPTAB(1) = INT2(0)
C
C     THE FOLLOWING WRITE STATEMENT PROVIDES HEADERS FOR THE
C     OUTPUT FILES.
C
```

```
      WRITE(14,30)'TIME',TAB,'TIME STEP',TAB,'ALTITUDE',TAB,
     &  'GRANGE',TAB,'WIND_AZIMUTH',TAB,'WIND_VEL',TAB,'LATITUDE',TAB,
     &  'LONGITUDE',TAB,'JUMP'
   30   FORMAT(2X,A,A1,A,A1,A,A1,A,A1,A,A1,A,A1,A,A1,A,A1,A)
C
C     WRITE THE INITIAL OUTPUT AT START TIME OF THE SIMULATION.
C
      WRITE(14,20)INIT_INTEGRAT_LMT,TAB,INTEGRAT_STEP,TAB,
     &    INITIAL_ALT,TAB,GRD_RANGE,TAB,WIND_AZIMUTH,TAB,
     &    WIND_VEL,TAB,INIT_LAT,TAB,INIT_LON,TAB,JMPTAB(1)
C
      INTEGRAT_STEP = FLIGHT_TIME / 2047.0
C
      IF (INTEGRAT_STEP .GT. 10000.0) THEN
        INTEGRAT_STEP = 10000.0
      ELSEIF (INTEGRAT_STEP .LT. 1.0) THEN
        INTEGRAT_STEP = 1.0
      ENDIF
C
      PERM_INT_STEP = INTEGRAT_STEP
      NEW_LAT = SNGL(INIT_LAT)
      NEW_LON = SNGL(INIT_LON)
C
C     READ IN ONE SET OF DATA POINTS OF THE ASCENT VELOCITY PROFILE.
C
      READ (21,*) INIT_SEG_TIME, INIT_SEG_VEL
  100 READ (21,*,END = 101) FIN_SEG_TIME, FIN_SEG_VEL
      SegmentsFinished = .FALSE.
      GO TO 102
  101 FIN_SEG_TIME = FLIGHT_TIME
      FIN_SEG_VEL  = 0.0
      INIT_SEG_VEL = 0.0
      SegmentsFinished = .TRUE.
  102 CONTINUE
C
C     COMPUTE THE SLOPE AND Y INTERCEPT USED IN THE FORMULA FOR THE
C     EQUATION OF A LINE.  THESE COMPONENTS COMPOSE THE FUNCTION USED
C     IN DEVELOPING THE PROFILE.
C
      SLOPE = ( FIN_SEG_VEL - INIT_SEG_VEL ) /
     &        ( FIN_SEG_TIME - INIT_SEG_TIME )
      Y_INTERCEPT = ( FIN_SEG_VEL - ( SLOPE * FIN_SEG_TIME ) )
      INIT_INTEGRAT_LMT = INIT_SEG_TIME
      FIN_INTEGRAT_LMT = INIT_SEG_TIME + INTEGRAT_STEP
C
      IF(DIFFGT .GT. TOLERANCE)THEN
        FIN_INTEGRAT_LMT  = DIFFGT + INIT_SEG_TIME
        INTEGRAT_STEP = DIFFGT
        DIFFGT = 0.0
      ENDIF
C
C     IF THE UPPER LIMIT, BASED ON THE TIMESTEP, IS GREATER THAN THE
C     FUNCTION BOUNDARY OF THE FUNCTION, RESET THE UPPER LIMIT TO
C     COMPENSATE FOR THE UPPER BOUNDARY.
C
      IF(FIN_INTEGRAT_LMT .GT. FIN_SEG_TIME)THEN
        DIFFGT = FIN_INTEGRAT_LMT - FIN_SEG_TIME
        FIN_INTEGRAT_LMT  = FIN_INTEGRAT_LMT - DIFFGT
        INTEGRAT_STEP = FIN_INTEGRAT_LMT - INIT_INTEGRAT_LMT
      ENDIF
C
      IF(FIN_INTEGRAT_LMT .GE. FLIGHT_TIME)THEN
        FIN_INTEGRAT_LMT  = FLIGHT_TIME
        INTEGRAT_STEP = FIN_INTEGRAT_LMT - INIT_INTEGRAT_LMT
        DIFFGT = 0.0
      ENDIF
C
C     USE FIRST TRAJECTORY INPUT AS 'RESET' VALUE.  THEN CHANGE MODE TO
C     'OPERATE' AND CYCLE THROUGH REMAINDER OF TRAJECTORY.
C
C     THE FUNINT (FUNCTION INTEGRATION) ROUTINE WILL PERFORM THE
C     INTEGRATION NEEDED TO PROVIDE THE DISTANCE TRAVELED (SEGMENT_DIST)
C     IN TIME DEFINED BY THE FUNCTION.
C
  200 IF(.NOT.SegmentsFinished)THEN
        CALL FUNINT(INIT_INTEGRAT_LMT,FIN_INTEGRAT_LMT,2,FUNCNT,ICHECK,
     &              SEGMENT_DIST,SLOPE,Y_INTERCEPT)
      ENDIF
C
C     COMPUTE THE CUMULATIVE DISTANCE TRAVELED BY THE BALLOON.
C
      TOTAL_SEG_DIST = TOTAL_SEG_DIST + SEGMENT_DIST
      FINAL_ALT = TOTAL_SEG_DIST + INITIAL_ALT
C
C     NOGAPS/GRAMRT DERIVES THE WIND COMPONENTS, EAST (UGH (M/S))
C     AND NORTH (VGH (M/S)).  THESE COMPONENTS ARE BASED UPON THE
C     LATITUDE (NEW_LAT (DEG)/LAT_RAD (RAD)), LONGITUDE (NEW_LON
C     (DEG)/LON_RAD (RAD)), AND ALTITUDE (FINAL_ALT (M)/HSIM (FEET)).
C
      IF(CLIMATE .EQ. 13)THEN
        CALL NOGAPS(FINAL_ALT, NEW_LAT, NEW_LON, UGH, VGH, FIRST)
      ELSE
        HSIM   = FINAL_ALT * 3.28084
        CALL GRAMRT (FIRST)
      ENDIF
C
C     CONVERT THE WIND COMPONENTS TO A DISTANCE.
C
      DIST_EAST = ((UGH + INIT_EAST_WIND)/2.0) * INTEGRAT_STEP
      DIST_NORTH = ((VGH + INIT_NORTH_WIND)/2.0) * INTEGRAT_STEP
C
      DIST_EAST = DIST_EAST + TOTAL_EAST
      DIST_NORTH = DIST_NORTH + TOTAL_NORTH
C
      TOTAL_EAST = DIST_EAST
      TOTAL_NORTH = DIST_NORTH
C
      INIT_EAST_WIND = UGH
      INIT_NORTH_WIND = VGH
C
      IF (VGH.EQ.0.0) THEN
        WIND_AZIMUTH = 90.0
      ELSE
        WIND_AZIMUTH = (ATAN(UGH/VGH))*R2D
      ENDIF
      IF( VGH.LT.0.0) THEN
        WIND_AZIMUTH = 180.0+WIND_AZIMUTH
      ELSEIF( UGH.LT.0.0 ) THEN
        WIND_AZIMUTH = 360.0+WIND_AZIMUTH
      ENDIF
C
      WIND_VEL = SQRT((UGH**2) + (VGH**2)
C
      DELTAT = INTEGRAT_STEP + DELTAT
C
      WCMAG = SQRT((DIST_EAST**2) + (DIST_NORTH**2))
C
C     THE LATITUDE/LONGITUDE RESOLUTION IS 1.0 METER (ANGLE_RESOLUTION).
```

```
C
      IF(WCMAG .GT. ANGLE_RESOLUTION)THEN
C
      TOTAL_EAST  = 0.0
      TOTAL_NORTH = 0.0
C
      LAT_DEG = LAT_RAD * R2D
      LON_DEG = LON_RAD * R2D
C
C     SUBROUTINE ECOORD USES THE LATITUDE (LAT_DEG), LONGITUDE (LON_DEG),
C     AS WELL AS THE DISTANCE THE BALLOON TRAVELED DURING THE TIMESTEP
C     (DIST_EAST, DIST_NORTH) TO PRODUCE THE NEW LATITUDE AND LONGITUDE
C     (NEW_LAT, NEW_LON).  IT ALSO USES THE CONSTANT RADIUS_ERTH AND THE FLAG
C     IS. THIS FLAG ALLOWS FOR A NORTH AND EAST RANGE INSTEAD OF
C     MAGNITUDE AND DIRECTION.
C
      CALL ECOORD (LAT_DEG,LON_DEG,DIST_EAST,DIST_NORTH,
     .             RADIUS_ERTH,IS,NEW_LAT,NEW_LON)
C
C     LON_ERTH_ROT  = (ERTH_ROTATION * DELTAT) * R2D
C     NEW_LON = NEW_LON + LON_ERTH_ROT
      IF (NEW_LON.GT.360.0) THEN
        NEW_LON = NEW_LON - 360.0
      ELSE IF (NEW_LON.LT.-360.0) THEN
        NEW_LON = NEW_LON + 360.0
      ENDIF
      DELTAT = 0.0
C
      DIFLAT = ABS(NEW_LAT - LAT_DEG)
      DIFLON = ABS(NEW_LON - LON_DEG)
C
      NEW_LAT = SNGL(NEW_LAT)
      NEW_LON = SNGL(NEW_LON)
      LAT_RAD = NEW_LAT/R2D
      LON_RAD = NEW_LON/R2D
      XLATSNG = SNGL(LAT_RAD)
      XLNGSNG = SNGL(LON_RAD)
C
      CURR_POSITION(1) = ((RADIUS_ERTH)*COS(XLATSNG))*SIN(XLNGSNG)
      CURR_POSITION(2) = ((RADIUS_ERTH)*COS(XLATSNG))*COS(XLNGSNG)
      CURR_POSITION(3) = (RADIUS_ERTH)*SIN(XLATSNG)
C
      CALL GRNRGE(INIT_POSITION,CURR_POSITION,AE,BE,RADIUS_ERTH,
     &            2,2,GRD_RANGE)
C
      ENDIF
C
      T2TOF = ABS(FIN_INTEGRAT_LMT - FLIGHT_TIME)
      PERCNT_DONE = JNINT((FIN_INTEGRAT_LMT/FLIGHT_TIME)*100.0)

      IF( PERCNT_DONE .GT. last_value) THEN
        WRITE(CHAR_DATA,4) PERCNT_DONE
    4   format ( i3 , ' %' )
        PRTEXT = CHAR_DATA
        CALL SetiText( %val(ItHndl), %val(PrText) )
        last_value = PERCNT_DONE
      ENDIF
C
      IF(T2TOF .LE. 0.5)GOTO 789
C
C     IF THE DIFFERENCE BETWEEN THE TIMESTEP AND SEGMENT IS GREATER THAN
C     ZERO, READ IN ANOTHER SEGMENT OF THE ASCENT PROFILE.
C
      IF( DIFFGT .GT. TOLERANCE )THEN
```

```
          INIT_SEG_TIME = FIN_SEG_TIME
          INIT_SEG_VEL  = FIN_SEG_VEL
          GOTO 100
        ENDIF
C
        IF( INTEGRAT_STEP .EQ. DIFFNC .AND. DIFFTM .GT. TOLERANCE) THEN
          INTEGRAT_STEP = DIFFTM
          INIT_SEG_TIME = FIN_SEG_TIME
          INIT_SEG_VEL  = FIN_SEG_VEL
          DIFFNC = 0.0
          DIFFTM = 0.0
          GOTO 100
        ENDIF
C
        INTEGRAT_STEP = PERM_INT_STEP
C
        NO_OF_PTS = NO_OF_PTS + 1
        IF(NO_OF_PTS .GT. 2048.0)GOTO 100
C
        TIME_ARRAY(NO_OF_PTS)      = FIN_INTEGRAT_LMT
        LAT_ARRAY(NO_OF_PTS)       = NEW_LAT
        LON_ARRAY(NO_OF_PTS)       = NEW_LON
        ALT_ARRAY(NO_OF_PTS)       = FINAL_ALT
        GRANGE_ARRAY(NO_OF_PTS)    = GRD_RANGE
        WINDAZ_ARRAY(NO_OF_PTS)    = WIND_AZIMUTH
        WIND_VEL_ARRAY(NO_OF_PTS)  = WIND_VEL
C
        TOFTAB(NO_OF_PTS) = FIN_INTEGRAT_LMT
        LATTAB(NO_OF_PTS) = NEW_LAT
        ALTTAB(NO_OF_PTS) = FINAL_ALT
C
C       CONVERT PROGRAM'S INTERNAL WEST LONGITUDE TO EAST LONGTUDE FOR MAP
        IF(NEW_LON.GT.180.0) THEN
          LNGTAB(NO_OF_PTS) = 360.0 - NEW_LON
          IF(OFF_WEST)THEN
            JMPTAB(NO_OF_PTS) = INT2(0)
          ELSE
            JMPTAB(NO_OF_PTS) = INT2(-1)
            OFF_WEST = .TRUE.
          ENDIF
        ELSE IF(NEW_LON.LT.-180.0) THEN
          LNGTAB(NO_OF_PTS) = 360.0 + NEW_LON
          IF(OFF_EAST)THEN
            JMPTAB(NO_OF_PTS) = INT2(0)
          ELSE
            JMPTAB(NO_OF_PTS) = INT2(1)
            OFF_EAST = .TRUE.
          ENDIF
        ELSE
          LNGTAB(NO_OF_PTS) = -1.0 * NEW_LON
          IF (OFF_WEST) THEN
            JMPTAB(NO_OF_PTS) = INT2(1)
            OFF_WEST = .FALSE.
          ELSE IF (OFF_EAST) THEN
            JMPTAB(NO_OF_PTS) = INT2(-1)
            OFF_EAST = .FALSE.
          ELSE
            JMPTAB(NO_OF_PTS) = INT2(0)
          ENDIF
        ENDIF
C
        WRITE(14,20)FIN_INTEGRAT_LMT,TAB,INTEGRAT_STEP,TAB,
       $              FINAL_ALT,TAB,GRD_RANGE,TAB,WIND_AZIMUTH,TAB,WIND_VEL,
       $              TAB,NEW_LAT,TAB,NEW_LON,TAB,JMPTAB(NO_OF_PTS)
```

```
   20 FORMAT (F9.2,A1,F8.2,A1,F10.3,A1,F15.3,A1,2(F8.3,A1),
     &          F7.4,A1,F8.4,A1,I2)
C
C     DETERMINE THE POSITION OF THE CUMULATIVE TIMESTEP ITERATIONS
C     IN RELATION TO THE END POINT OF THE ASCENT PROFILE SEGMENT.
C
      DIFFNC = ABS(FIN_INTEGRAT_LMT - FIN_SEG_TIME)
      DIFFTM = ABS(DIFFNC - INTEGRAT_STEP)
C
      IF( DIFFNC .LT. TOLERANCE )THEN
        INIT_SEG_TIME = FIN_SEG_TIME
        INIT_SEG_VEL  = FIN_SEG_VEL
        GOTO 100
      ELSEIF( DIFFNC .GE. INTEGRAT_STEP .OR. DIFFTM .LT.TOLERANCE )THEN
        INIT_INTEGRAT_LMT = FIN_INTEGRAT_LMT
        FIN_INTEGRAT_LMT  = FIN_INTEGRAT_LMT + INTEGRAT_STEP
        GO TO 200
      ELSE
        INTEGRAT_STEP     = DIFFNC
        INIT_INTEGRAT_LMT = FIN_INTEGRAT_LMT
        FIN_INTEGRAT_LMT  = FIN_SEG_TIME
        GOTO 200
      ENDIF
C
  789 NO_OF_PTS = NO_OF_PTS + 1
C
      TIME_ARRAY(NO_OF_PTS)     = FIN_INTEGRAT_LMT
      LAT_ARRAY(NO_OF_PTS)      = NEW_LAT
      LON_ARRAY(NO_OF_PTS)      = NEW_LON
      ALT_ARRAY(NO_OF_PTS)      = FINAL_ALT
      GRANGE_ARRAY(NO_OF_PTS)   = GRD_RANGE
      WINDAZ_ARRAY(NO_OF_PTS)   = WIND_AZIMUTH
      WIND_VEL_ARRAY(NO_OF_PTS) = WIND_VEL
C
      ntrpts = NO_OF_PTS
      TOFTAB(NO_OF_PTS) = FIN_INTEGRAT_LMT
      LATTAB(NO_OF_PTS) = NEW_LAT
      ALTTAB(NO_OF_PTS) = FINAL_ALT
C
C     CONVERT PROGRAM'S INTERNAL WEST LONGITUDE TO EAST LONGTUDE FOR MAP
      IF(NEW_LON.GT.180.0) THEN
        LNGTAB(NO_OF_PTS) = 360.0 - NEW_LON
        IF(OFF_WEST)THEN
          JMPTAB(NO_OF_PTS) = INT2(0)
        ELSE
          JMPTAB(NO_OF_PTS) = INT2(-1)
          OFF_WEST = .TRUE.
        ENDIF
      ELSE IF(NEW_LON.LT.-180.0) THEN
        LNGTAB(NO_OF_PTS) = 360.0 + NEW_LON
        IF(OFF_EAST)THEN
          JMPTAB(NO_OF_PTS) = INT2(0)
        ELSE
          JMPTAB(NO_OF_PTS) = INT2(1)
          OFF_EAST = .TRUE.
        ENDIF
      ELSE
        LNGTAB(NO_OF_PTS) = -1.0 * NEW_LON
        IF (OFF_WEST) THEN
          JMPTAB(NO_OF_PTS) = INT2(1)
          OFF_WEST = .FALSE.
        ELSE IF (OFF_EAST) THEN
          JMPTAB(NO_OF_PTS) = INT2(-1)
          OFF_EAST = .FALSE.
        ELSE
          JMPTAB(NO_OF_PTS) = INT2(0)
        ENDIF
      ENDIF
C
      call SetPort( %val(SavedPort) )
      call DisposDialog( %val(StatPtr) )
C
C     CLOSE TEMP FILES
C
      CLOSE(UNIT=IOTEM1,STATUS='DELETE')
      CLOSE(UNIT=IOTEM2)
C
C     CLOSE OUTPUT FILE
C
      CLOSE(UNIT=14)
C
      RETURN
      END
C
      FUNCTION FUNCNT(T,SLOPE,YINTER)
      REAL*4 SLOPE,YINTER,FUNCNT
      REAL*8 T
C
      FUNCNT = ( SLOPE * T ) + YINTER
C
      RETURN
      END
C
      SUBROUTINE FUNINT (XL,XH,INTORD,FNAM,ICHECK,VOINT,P,Y)
C
C**********************************************************************C
C                                                                      C
C   PURPOSE  1) A FUNCTIONAL INTEGRATION ROUTINE USING A VARIABLE      C
C                   ORDER GAUSSIAN-LEGENDRE ALGORITHM.                 C
C                                                                      C
C   INPUTS   DESCRIPTION                                               C
C                                                                      C
C       ( XL )   THE LOWER BOUND ON THE INTEGRAL (CAN BE REAL*8)       C
C       ( XH )   THE UPPER BOUND ON THE INTEGRAL (CAN BE REAL*8)       C
C   (INTORD) ORDER OF THE GAUSS-LEGENDRE POLYNOMIAL: 2-10              C
C                INCLUSIVE AND 12, 16, 20, 24 AND 32 ARE IMPLEMENTED.  C
C       (FNAM)   NAME OF THE FORTRAN EXTERNAL FUNCTION                 C
C                TO BE INTEGRATED (CAN BE REAL*8)                      C
C                                                                      C
C       OUTPUTS  DESCRIPTION                                           C
C                                                                      C
C   (VOINT)  THE VALUE OF THE INTEGRAL BETWEEN THE SPECIFIED LIMITS    C
C   (ICHECK) CHECK FOR: PROPER EXECUTION = 0                           C
C               FAULT FOUND      = -1                                  C
C                                                                      C
C   PROGRAM REQUIREMENTS                                               C
C                                                                      C
C   THE PROGRAM ASSUMES THAT A FORTRAN EXTERNAL FUNCTION (WHICH        C
C       MAY BE REAL*8) IS DEFINED SOMEWHERE WITHIN THE CALLING CODE.   C
C   ( SINCE IT IS AN EXTERNAL FUNCTION IT NEED NOT NECESSARILY BE      C
C   DEFINED IN THE ROUTINE THAT MAKES THE SUBROUTINE CALL ).           C
C                                                                      C
C       CAVEATS                                                        C
C                                                                      C
C       THIS IS A VERY SIMPLE ROUTINE WHICH NEVERTHELESS IS VERY       C
C       FAST (THERE ARE ONLY INTORD EVALUATIONS OF THE FUNCTION)       C
C       AND QUITE ACCURATE FOR WELL BEHAVED FUNCTIONS.                 C
C       AS WITH ANY NUMERICAL TECHNIQUE FUNCTIONS WITH SINGULARITIES   C
```

```
C          AND PERIODIC FUNCTIONS INTEGRATED OVER MANY PERIODS ARE OFTEN  C
C          NOT HANDLED WELL.                                               C
C          IN PRINCIPLE THE ACCURACY OF THE ALGORITHM INCREASES WITH THE  C
C          THE ORDER OF THE POLYNOMIAL USED, HOWEVER ROUNDOFF ERRORS WILL C
C          LIMIT THIS TREND.                                               C
C                                                                          C
C--------------------------------------------------------------------------C
C
        PARAMETER (MIN = 2, MAX = 32)
        REAL*8 WEIGHT(MIN:MAX,MAX/2), Z(MIN:MAX,MAX/2)
        REAL*8 SUMH, DELTAH, HALF, TERM
        REAL*4 VOINT, XH, XL, FNAM
        CHARACTER*1 SR
        DATA HALF / 0.5 /
C
C       DATA STATEMENTS CONTAINING THE WEIGHTS AND ZEROES OF THE
C       LEGENDRE POLYNOMIALS
C
        DATA WEIGHT(2,1) / 1. /
        DATA (WEIGHT(3,I),I=1,2) / .555555555555556,.888888888888889 /
        DATA (WEIGHT(4,I),I=1,2) / .347854845137454,.652145154862546 /
        DATA (WEIGHT(5,I),I=1,3) / .236926885056189,.478628670499366,
     &      .568888888888889 /
        DATA (WEIGHT(6,I),I=1,3) / .171324492379170,.360761573048139,
     &      .467913934572691 /
        DATA (WEIGHT(7,I),I=1,4) / .129484966168870,.279705391489277,
     &      .381830050505119,.417959183673469 /
        DATA (WEIGHT(8,I),I=1,4) / .101228536290376,.222381034453374,
     &      .313706645877887,.362683783378362 /
        DATA (WEIGHT(9,I),I=1,5) / .081274388361574,.180648160694857,
     &      .260610696402935,.312347077040003,.330239355001260 /
        DATA (WEIGHT(10,I),I=1,5) / .066671344308688,.149451349150581,
     &      .219086362515982,.269266719309996,.295524224714753 /
        DATA (WEIGHT(12,I),I=1,6) / .047175336386512,.106939325995318,
     &      .160078328543346,.203167426723066,.233492536538355,
     &      .249147045813403 /
        DATA (WEIGHT(16,I),I=1,8) / .027152459411754,.062253523938648,
     &      .095158511682493,.124628971255534,.149595988816577,
     &      .169156519395003,.182603415044924,.189450610455068 /
        DATA (WEIGHT(20,I),I=1,10) / .017614007139152,.040601429800387,
     &      .062672048334109,.083276741576705,.101930119817240,
     &      .118194531961518,.131688638449177,.142096109318382,
     &      .149172986472604,.152753387130726 /
        DATA (WEIGHT(24,I),I=1,12) / .012341229799987,.028531388628934,
     &      .044277438817420,.059298584915437,.073346481411080,
     &      .086190161531953,.097618652104114,.107444270115966,
     &      .115505668053726,.121670472927803,.125837456346828,
     &      .127938195346752 /
        DATA (WEIGHT(32,I),I=1,16) / .007018610009470,.016274394730906,
     &      .025392065309262,.034273862913021,.042835898022227,
     &      .050998059262376,.058684093478536,.065822222776362,
     &      .072345794108849,.078193895787070,.083311924226947,
     &      .087652093004404,.091173878695764,.093844399080805,
     &      .095638720079274,.096540088514728 /
C
        DATA Z(2,1)           / .577350269189626 /
        DATA (Z(3,I),I=1,2) / .774596669241483,.0 /
        DATA (Z(4,I),I=1,2) / .861136311594053,.339981043584856 /
        DATA (Z(5,I),I=1,3) / .906179845938664,.538469310105683,.0 /
        DATA (Z(6,I),I=1,3) / .932469514203152,.661209386466265,
     &      .238619186083197 /
        DATA (Z(7,I),I=1,4) / .949107912342759,.741531185599394,
     &      .405845151377397,.0 /
        DATA (Z(8,I),I=1,4) / .960289856497536,.796666477413627,
     &     .525532409916329,.183434642495650 /
      DATA (Z(9,I),I=1,5) / .968160239507626,.836031107326636,
     &     .613371432700590,.324253423403809,.0 /
      DATA (Z(10,I),I=1,5) / .973906528517172,.865063366688985,
     &     .679409568299024,.433395394129247,.148874338981631 /
      DATA (Z(12,I),I=1,6) / .981560634246719,.904117256370475,
     &     .769902674194305,.587317954286617,.367831498998180,
     &     .125233408511469 /
      DATA (Z(16,I),I=1,8) / .989400934991650,.944575023073233,
     &     .865631202387832,.755404408355003,.617876244402644,
     &     .458016777657227,.281603550779259,.095012509837637 /
      DATA (Z(20,I),I=1,10) / .993128599185095,.963971927277914,
     &     .912234428251326,.839116971822219,.746331906460151,
     &     .636053680726515,.510867001950827,.373706088715420,
     &     .227785851141645,.076526521133497 /
      DATA (Z(24,I),I=1,12) / .995187219997021,.974728555971309,
     &     .938274552002733,.886415527004401,.820001985973903,
     &     .740124191578554,.648093651936976,.545421471388840,
     &     .433793507626045,.315042679696163,.191118867473616,
     &     .064056892862686 /
      DATA (Z(32,I),I=1,16) / .997263861849481,.985611511545268,
     &     .964762255587506,.934906075937740,.896321155766052,
     &     .849367613732570,.794483795967942,.732182118740290,
     &     .663044266930215,.587715757240762,.506899908932229,
     &     .421351276130635,.331868602282128,.239287362252137,
     &     .144471961582796,.048307665687738 /
C
      VOINT  = 0.
      ICHECK = 0
C
C     CHECK THAT THE ORDER OF INTEGRATION IS IN BOUNDS
C
      IF ( (INTORD .LT. MIN) .OR. (INTORD .GT. MAX) ) THEN
        PRINT*, ' INTEGRATION OF THAT ORDER IS NOT AVAILABLE. '
        ICHECK = -1
        RETURN
      END IF
C
C     CHECK THAT THE ORDER OF INTEGRATION IS AVAILABLE
C
      IF ( (WEIGHT(INTORD, 1) .EQ. 0.) ) THEN
        PRINT*, ' INTEGRATION OF THAT ORDER IS NOT AVAILABLE. '
        ICHECK = -1
        RETURN
      END IF
C
C     CALCULATE SEVERAL NECESSARY QUANITIIES
C
      DELTAH  = (XH - XL) * HALF
      SUMH    = (XH + XL) * HALF
      NSTEP = INTORD/2
C
C     INTEGRATE THE FUNCTION WITH A INTORD ORDER ALGORITHM
C
      DO I = 1,NSTEP
        TERM = FNAM ( SUMH + DELTAH * Z (INTORD,I),F,Y)
        VOINT = VOINT + WEIGHT (INTORD,I) * TERM
        TERM = FNAM ( SUMH - DELTAH * Z (INTORD,I),F,Y)
        VOINT = VOINT + WEIGHT (INTORD,I) * TERM
      END DO
C
C     MULTIPLY BY LIMITS CONSTANT AND TAKE CARE OF ODD POINT FOR INTORD ODD
C
      IF ( NSTEP*2 .EQ. INTORD ) THEN
```

```
        VOINT = DELTAH * VOINT
      ELSE
        VOINT = ( VOINT + WEIGHT (INTORD, NSTEP + 1)*FNAM(SUMH,P,Y))*
     &            DELTAH
      END IF
C
      RETURN
      END
C
      SUBROUTINE ECOORD ( ALATR, ALONR, DRP, URP, RE, IS, ALATP, ALONP )
C
C======================================================================C
C                                                                      C
C     PURPOSE   1) DETERMINE LATITUDE (ALATP) AND LONGITUDE (ALONP)   C
C                  OF A DESIRED POINT USING REFERENCE LATITUDE        C
C                  (ALATR) AND LONGITUDE (ALONR).                      C
C               2) INPUTS  ARE INTERPRETED DIFFERENTLY BY VALUE OF     C
C                  FLAG (IS)                                           C
C                    (IS) = 0  (DRP) RANGE FROM REFERENCE POINT        C
C                              (+) EASTERLY TO MERIDIAN OF DESIRED     C
C                              POINT                                   C
C                              (URP) RANGE ALONG MERIDIAN (+) UP TO    C
C                              DESIRED POINT                           C
C                    (IS) = 1  (DRP) RANGE FROM REFERENCE POINT TO     C
C                              DESIRED POINT                           C
C                              (URP) AZIMUTH OF RANGE LINE-OF-SITE TO  C
C                              DESIRED POINT                           C
C                                                                      C
C                    NOTE - RANGES ARE ALONG EARTH SURFACE AND         C
C                           AZIMUTH IS IN DEGREES FROM NORTH           C
C                           LATITUDES ARE GEOCENTRIC                   C
C                                                                      C
C     INPUTS    DESCRIPTION                                            C
C                                                                      C
C     (ALATR)   LATITUDE OF REFERENCE POINT (DEG)                      C
C     (ALONR)   LONGITUDE (+) EAST OF GREENWICH OF REFERENCE POINT     C
C               (DEG)                                                  C
C     (DRP)     RANGE FROM REFERENCE POINT TO                          C
C                  (IS) = 0 (+) EAST ALONG LATITUDE TO MERIDIAN OF     C
C                           DESIRED POINT                              C
C                  (IS) = 1 DESIRED POINT                              C
C     (URP)     (IS) = 0 RANGE (+) NORTH ALONG MERIDIAN OF DESIRED     C
C                        POINT FROM REFERENCE LATITUDE TO DESIRED      C
C                        POINT                                         C
C               (IS) = 1 AZIMUTH OF RANGE LINE-OF-SITE FROM REFERENCE  C
C                        POINT TO DESIRED POINT                        C
C     (RE)      EARTH RADIUS AT REFERENCE POINT                        C
C     (IS)      FLAG TO INTERPRET THE MEANINGS OF (DRP) AND (URP)      C
C                                                                      C
C     OUTPUTS   DESCRIPTION                                            C
C                                                                      C
C     (ALATP)   LATITUDE OF DESIRED POINT (DEG)                        C
C     (ALONP)   LONGITUDE OF DESIRED POINT (DEG)                       C
C                                                                      C
C   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -  C
C                                                                      C
C     REQUIREMENTS 1) SINE ROUTINE        - SIN                        C
C                  2) COSINE ROUTINE      - COS                        C
C                  3) SQUARE ROOT ROUTINE - SQRT                       C
C                  4) ARCSINE ROUTINE     - ASIN                       C
C                                                                      C
C======================================================================C
C
      DATA R2D / 57.29577951 /
      REAL*4 ALATR,ALONR,DRP,URP,ALATP,ALONP
C
      ALONR = ALONR * -1.0
C
      IF( RE ) 50, 50, 10
   10 CONTINUE
      THD = DRP/RE
C
      IF( IS ) 20, 20, 30
C
   20 CONTINUE
      THU = URP/RE
      TT  = COS ( THU )
      T1  = SIN ( THU )
      T2  = COS ( THD )*TT
      T3  = SIN ( THD )*TT
      GO TO 40
C
   30 CONTINUE
      THU = URP/R2D
      TT  = SIN ( THD )
      T1  = COS ( THU )*TT
      T2  = COS ( THD )
      T3  = SIN ( THU )*TT
C
   40 CONTINUE
      TT  = ALATR/R2D
      SPP = COS ( TT )*T1 + SIN ( TT )*T2
      IF( SPP .GT.  1.0 ) SPP = 1.
      IF( SPP .LT. -1.0 ) SPP =-1.
      SLP = SQRT ( 1. - SPP*SPP )
      IF( SLP .NE.  0.0 ) SLP = T3/SLP
      IF( SLP .LT. -1.0 ) SLP =-1.
      IF( SLP .GT.  1.0 ) SLP = 1.
C
      ALATP = ASIN ( SPP )*R2D
      ALONP = ASIN ( SLP )*R2D + ALONR
   50 CONTINUE
C
      ALONR = ALONR * -1.0
      ALONP = ALONP * -1.0
      RETURN
      END
C
      SUBROUTINE OPENIT(XLATIN,XLONIN,INFILE,TOTTOF,ICLIMT)
C
C.... THIS ROUTINE OPENS THE FILES NORMALLY ATTACHED WITH JCL DIRECTIVES
C     ON THE GOULD VERSION
C
C.... WRITTEN   10/18/89   L SCHILLING   NASA/ADFRF.
C
      CHARACTER*255 INFILE
      CHARACTER CLAT*3,CLON*4,CLIMT*2
      CHARACTER*7 CTOF
      INTEGER LAT,LON,TOF,ICLIMT
C
      LAT = XLATIN
      LON = XLONIN
      TOF = TOTTOF
      WRITE(CLAT,'(I3)')LAT
      WRITE(CLON,'(I4)')LON
      WRITE(CTOF,'(I7)')TOF
      WRITE(CLIMT,'(I2)')ICLIMT
C
```

```
      OPEN (UNIT=21,FILE=INFILE,STATUS='OLD',READONLY)
C
      OPEN(14,FILE=CLAT//'_'//CLON//'_'//CTOF//'.'//CLIMT,
     .     FORM='FORMATTED',ACCESS='SEQUENTIAL', STATUS='NEW')
C
      RETURN
      END

C.... THIS FILE CONTAINS SUBROUTINES USED BY REAL-TIME GRAM PROGRAM.
C     1/24/89   L SCHILLING.  THE MAJORITY HAVE BEEN UNAFFECTED, BUT
C     SEVERAL HAVE BEEN SIGNIFICANTLY MODIFIED FROM THE ORIGINAL
C     GRAM88 CODE.
C
      SUBROUTINE CORLAT(A,B,C,D,E,F,G,H,AI,AJ,AK,SP1,SP2,SD1,SD2,ST1,
     1 ST2,SU1,SU2,SV1,SV2,UD1,UD2,VD1,VD2,RD,RT,RV)
      IF(SD1*ST1*SD2*ST2*RD*RT*RV.GT.0.) GO TO 5
C.....DEFAULT VALUES AVOID DIVISION BY ZERO
      IF(SD1.LE.0.) SD1=0.001
      IF(ST1.LE.0.) ST1=0.001
      IF(SD2.LE.0.) SD2=0.001
      IF(ST2.LE.0.) ST2=0.001
      IF(RD.LE.0.) RD = .00001
      IF(RT.LE.0.) RT = .00001
      IF(RV.LE.0.) RV = .00001
    5 CONTINUE
      IF (ABS(UD1).LE.0.) UD1 = 0.001
      IF (ABS(VD1).LE.0.) VD1 = 0.001
      IF (ABS(SU1).LE.0.) SU1 = 0.001
      IF (ABS(SV1).LE.0.) SV1 = 0.001
      IF (ABS(UD1).GE.1.) UD1 = 0.99*UD1/ABS(UD1)
      IF (ABS(VD1).GE.1.) VD1 = 0.99*VD1/ABS(VD1)
      A=RD*SD2/SD1
      B=SD2*SQRT(1-RD*RD)
      TD2=(SP2*SP2-SD2*SD2-ST2*ST2)/(2*SD2*ST2)
      TD1=(SP1*SP1-SD1*SD1-ST1*ST1)/(2*SD1*ST1)
      IF(ABS(TD1).LE.0.) TD1 = 0.001
      IF(ABS(TD2).LE.0.) TD2 = 0.001
      IF(ABS(TD2).GE.1.0) TD2=0.99*TD2/ABS(TD2)
      IF(ABS(TD1).GE.1.0) TD1=0.99*TD1/ABS(TD1)
      C=(ST2/ST1)*(RT-RD*TD2*TD1)/(1-TD1*TD1*RD*RD)
      D=(RT*ST2-C*ST1)/(A*TD1*SD1)
      E=      ST2*ST2-C*C*ST1*ST1-D*D*SD2*SD2-2*C*D*RD*TD1*ST1*SD2
      IF(E.GE.0.) GO TO 10
      E=0.
   10 E=SQRT(E)
      F=(SU2/SU1)*(RV-RD*UD2*UD1)/(1-RD*RD*UD1*UD1)
      G=(RV*SU2-F*SU1)/(RD*UD1*SD2)
      H=      SU2*SU2-F*F*SU1*SU1-G*G*SD2*SD2-2*F*G*RD*UD1*SD2*SU1
      IF(H.GE.0.) GO TO 15
      H=0.
   15 H=SQRT(H)
      AI=(SV2/SV1)*(RV-RD*VD2*VD1)/(1-RD*RD*VD1*VD1)
      AJ=(RV*SV2-AI*SV1)/(RD*VD1*SD2)
      AK=      SV2*SV2-AI*AI*SV1*SV1-AJ*AJ*SD2*SD2-2*AI*AJ*RD*VD1*SD2*SV1
      IF(AK.GE.0.) GO TO 25
      AK=0.
   25 AK=SQRT(AK)
      RETURN
      END
C
      FUNCTION CORREL(X)
      DATA A,B/19.51615854016301,1.000416939412455578/
      RHO = 1./EXP(B*X)
      IF(X.LT.0.05)RHO = 1. - A*X**2
      CORREL = RHO
      RETURN
      END
C
      SUBROUTINE FAIR (PG, DG, TG, PJ, DJ, TJ, IH, P, D, T, DPXG,
     $ DPYG, DPXJ, DPYJ, DPX, DPY, DTXG, DTYG, DTXJ, DTYJ, DTX, DTY)
C.....FAIRS BETWEEN GROVES AND JACCHIA VALUES 90 LE HEIGHT LE 115 KM
      DIMENSION CZ(6)
C.....FAIRING VALUES
      DATA CZ /1.0,0.9045085,0.6545085,0.3454915,0.0954915,0.0/
C     HEIGHT INDEX
      I =(IH - 85)/5
C     GROVES FAIRING COEFFICIENT
      CZI = CZ(I)
C     JACCHIA FAIRING COEFFICIENT
      SZI = 1.0 - CZI
C     FAIRED TEMPERATURE
      T = TG*CZI + TJ*SZI
C     FAIRED DENSITY
      D = EXP(ALOG(DG)*CZI + ALOG(DJ)*SZI)
C     FAIRED GAS CONSTANT
      RG = PG/(DG*TG)
      RJ = PJ/(DJ*TJ)
      R = RG*CZI + RJ*SZI
      P = R*D*T
      DPX = DPXG*CZI + DPXJ*SZI
C     DP/DY FOR GEOSTROPHIC WINDS
      DPY=DPYG*CZI+DPYJ*SZI
      DTX = DTXG*CZI + DTXJ*SZI
C      DT/DY FOR THERMAL WINDS
      DTY = DTYG * CZI + DTYJ * SZI
      RETURN
      END
C
      SUBROUTINE GRAMIN (XLATIN,XLONIN,ALTINT,ICLIMT)
C
C.... THIS ROUTINE READS IN ALL DATA FILES TO INITIALIZE ARRAYS
C     FOR USE IN GRAM PROGRAM.
C
C.... WRITTEN   24 JAN 89   L SCHILLING   NASA/ADFRF.
C
      COMMON /C4    /  DUMMY4(2529), THET1, DUMMY5(2)
      COMMON /IOTEMP/ IOTEM1,IOTEM2,IUG   ,IUN   ,DD    ,XMJD  ,PHI1   ,
     .                PHI   ,NSAME ,RP1   ,RD1   ,RT1   ,SP1   ,SD1    ,
     .                ST1   ,RU1   ,RV1   ,SU1   ,SV1   ,MN    ,IDA    ,
     .                IYR   ,H1    ,PHI1R ,THET1R,G     ,RI    ,H      ,
     .                PHIR  ,THETR ,F10   ,F10B  ,AP    ,IHR   ,MIN    ,
     .                NMORE ,DX    ,HL    ,VL    ,DZ    ,B     ,EPS    ,
     .                IOPP  ,LOOK  ,IET   ,GLAT  ,RP1S  ,RD1S  ,RT1S   ,
     .                RU1S  ,RV1S  ,SP1S  ,SD1S  ,ST1S  ,SU1S  ,SV1S   ,
     .                UDS1  ,VDS1  ,UDL1  ,VDL1  ,UDS2  ,VDS2  ,UDL2   ,
     .                VDL2  ,REARTH
      COMMON /CHIC  / LA(4,4),NB(2),IWSYM,UCOEF(14,9),VCOEF(14,9)
      COMMON /WINCOM/ DUMSTF(17),UPRE,VPRE,DUPRE,DVPRE
C
      DATA PI  /3.141593  /
      DATA FAC /0.01745329/
C
      LOOK  = 0
      H     = 0.0
      NSAME = 0
C
C.... FIRST READ DEFINES INITIAL HEIGHT (KM), INITIAL LATITUDE (DEG)
C     INITIAL LONGITUDE (DEG), F10.7, MEAN F10.7, AP, MONTH, DAY,
C     YEAR (TOTAL YEAR - 1900), GREENWICH HOUR, MINUTES, SECONDS,
C     LATITUDE INCREMENT (DEG), LONGITUDE INCREMENT (DEG),
C     HEIGHT DECREASE (KM), MAXIMUM NUMBER OF POSITIONS (EXCLUDING
C     INITIAL POSITION) TO BE COMPUTED, TIME INCREMENT BETWEEN
C     POSITIONS, TRAJECTORY OPTION, OUTPUT OPTION, MINIMUM GEOSTROPHIC
C     LATITUDE.
C
C... SET INITIAL CONDITIONS FOR INITIALIZATION PROCESS
      H1    = ALTINT/1000.
      PHI1  = XLATIN
      THET1 = XLONIN
      MN    = ICLIMT
C... READ FIXED INPUT DATA
      OPEN(UNIT=55,FILE='FIXED_INPUT.',STATUS='OLD',READONLY)
      READ(55,*,END=90) F10   ,F10B  ,AP    ,
     .                  IDA   ,IYR   ,IHRO  ,MINO  ,ISECO ,
     .                  DPHI  ,DTHET ,DH    ,NMAX  ,INCT  ,IOPT  ,
     .                  IOPP  ,GLAT
C
      GLAT = ABS(GLAT)
      IF (GLAT.LT. 5.) GLAT =  5.
      IF (GLAT.GE.18.) GLAT = 17.999
      GLATF=GLAT*FAC
C
C.... INITIALIZE DATA ARRAYS.
C
      CALL SETUP
C      CALL GRIDIN
C
      CLOSE(55)
      RETURN
C
   90 CONTINUE
C
      WRITE(6,555)
  555 FORMAT('  GRAMIN PROBLEM')
      STOP
      END
C
      SUBROUTINE GRAMRT (FIRST)
C
C.... THIS ROUTINE IS THE EXECUTIVE FOR THE REAL-TIME GRAM PROGRAM.
C
C.... WRITTEN   26 JAN 89   L SCHILLING   NASA/ADFRF.
C
C.... GRAM INPUTS:
C                           GRAM            DRYDEN SIM
C                        -------------    --------------
C        TIME            SEC              SEC
C        ALTITUDE        KILOMETERS       FEET
C        LATITUDE        DEG, +NORTH      DEG, +NORTH
C        LONGITUDE       DEG, +WEST       DEG, +EAST
C
C.... GRAM OUTPUTS:
C
C                  UNPERTURBED (MONTHLY MEAN)
C        ===================================================
C                             GRAM              DRYDEN SIM
C                          -------------      --------------
C        PRESSURE          NEWTONS/METER**2   POUNDS/FT**2
C        DENSITY           KILOGRAMS/METER**3 SLUGS/FT**3
C        TEMPERATURE       DEGREES KELVIN     DEGREES RANKINE
C        GEOSTROPHIC WIND  METERS/SEC         FEET/SEC
C
```

```
C                     MEAN PLUS PERTURBATIONS
C        ======================================================
C                               GRAM                 DRYDEN SIM
C                            -------------        --------------
C        PRESSURE            NEWTONS/METER**2       POUNDS/FT**2
C        DENSITY             KILOGRAMS/METER**3     SLUGS/FT**3
C        TEMPERATURE         DEGREES KELVIN         DEGREES RANKINE
C        TOTAL WIND          METERS/SEC             FEET/SEC
C
C        THERMAL WIND SHEAR  METERS/SEC/KILOMETER   FEET/SEC/FOOT
C
      LOGICAL FIRST
C
C.... GRAM PROGRAM COMMON BLOCKS.
C
      COMMON /C4     /
     .                GLAT(16),GLON(16),NG    ,P4D(16,26),D4D(16,26),
     .                T4D(16,26),SP4(16,26),SD4(16,26),ST4(16,26),
     .                THET1 ,THET  ,HS
      COMMON /CHIC  / LA(4,4),NB(2),IWSYM,UCOEF(14,9),VCOEF(14,9)
      COMMON /COMJAC/ LAT_RADJ ,XLONG ,SDA   ,SHA   ,DY    ,R88   ,TE     ,
     .                EM
      COMMON /COMPER/ SPH   ,SDH   ,STH   ,PRH   ,DRH   ,TRH   ,URH   ,
     .                VRH   ,SUH   ,SVH   ,CP    ,PRHS  ,DRHS  ,TRHS  ,
     .                URHS  ,VRHS  ,PRHL  ,DRHL  ,TRHL  ,URHL  ,VRHL  ,
     .                SPHS  ,SDHS  ,STHS  ,SUHS  ,SVHS  ,SPHL  ,SDHL  ,
     .                STHL  ,SUHL  ,SVHL
      COMMON /IOTEMP/ IOTEM1,IOTEM2,IUG   ,IUN   ,DD    ,XMJD  ,PHI1  ,
     .                PHI   ,NSAME ,RP1   ,RD1   ,RT1   ,SP1   ,SD1   ,
     .                ST1   ,RU1   ,RV1   ,SU1   ,SV1   ,MN    ,IDA   ,
     .                IYR   ,H1    ,PHI1R ,THET1R,G     ,RI    ,H     ,
     .                PHIR  ,THETR ,F10   ,F10B  ,AP    ,IHR   ,MIN   ,
     .                NMORE ,DX    ,HL    ,VL    ,DZ    ,B     ,EPS   ,
     .                IOPP  ,LOOK  ,IET   ,GLATX ,RP1S  ,RD1S  ,RT1S  ,
     .                RU1S  ,RV1S  ,SP1S  ,SD1S  ,ST1S  ,SU1S  ,SV1S  ,
     .                UDS1  ,VDS1  ,UDL1  ,VDL1  ,UDS2  ,VDS2  ,UDL2  ,
     .                VDL2  ,REARTH
      COMMON /IPRTP / IPRT
      COMMON /PDTCOM/
     .                IU4   ,MONTH ,IOPR  ,PG(18,19),TG(18,19),
     .                DG(18,19),PSP(8,10,12),DSP(8,10,12),TSP(8,10,12),
     .                PAQ(17,5),DAQ(17,5),TAQ(17,5),PDQ(17,5),DDQ(17,5),
     .                TDQ(17,5),PR(20,10),DR(20,10),TR(20,10),UAQ(17,5),
     .                VAQ(17,5),UDQ(17,5),VDQ(17,5),UR(25,10),VR(25,10),
     .                PQ    ,DQ    ,TQ    ,UQ    ,VQ    ,PQA   ,DQA   ,
     .                TQA   ,UA    ,VA    ,IOPQ  ,PLP(25,10),DLP(25,10),
     .                TLP(25,10),ULP(25,10),VLP(25,10),UDL(25,10),
     .                VDL(25,10),UDS(25,10),VDS(25,10)
      COMMON /WINCOM/ DH    ,FCORY ,DX5   ,DY5   ,DPX   ,DPY   ,DPXX  ,
     .                DPXY  ,DPYY  ,UGH   ,VGH   ,TH    ,DTX   ,DTY   ,
     .                DUH   ,DVH   ,PH    ,UPRE  ,VPRE  ,DUPRE ,DVPRE
C
C.... COMMON BLOCKS ADDED IN MODIFYING GRAM AND INTERFACING WITH SIM.
C
      COMMON /GRAMOT/ PGH   ,DGH   ,TGH   ,UH    ,VH    ,PS    ,DS    ,
     .                TS    ,PGHP  ,DGHP  ,TGHP  ,PHP   ,DHP   ,THP   ,
     .                PSH   ,DSH   ,TSH   ,WGH
      LOGICAL         GRMATM,G76ATM,GATMP ,GRMWND,GWINDP
      COMMON /GRMDAT/ GRMATM,G76ATM,GATMP ,GRMWND,GWINDP,CS76  ,CSU   ,
     .                CSP   ,TMPP76,TMPRU ,TMPRP ,PA76  ,PAU   ,PAP   ,
     .                RHO76 ,RHOU  ,RHOP  ,UWINDU,UWINDP,VWINDU,VWINDP,
     .                USHEAR,VSHEAR
      COMMON /NASPGM/ PDAT(5720) ,DDAT(5720) ,TDAT(5720) ,
     .                SPDAT(5720),SDDAT(5720),STDAT(5720)
```

```
      LOGICAL         WIND
      COMMON /WINDAT/ WIND  ,XWIND ,YWIND ,NUMWND,ALTW(32),VELW(32),
     .                HDGW(32),XWA(32),YWA(32)
C
C.... DRYDEN SIM COMMON BLOCKS.
C
      COMMON /ALTFUN/ ASIM  ,RHO   ,GSIM  ,PA    ,TMPR
      REAL*8 FSIM, T, XLAT, XLNG
      COMMON /DRVOUT/ FSIM(13),DFSIM(13)
      COMMON /DRVOT2/ ALP   ,ALPDOT,BTA   ,BTADOT,HSIM  ,HDOT  ,
     .                V     ,VDOT  ,X     ,XDOT  ,Y     ,YDOT  ,VI
      LOGICAL         OP,RST,HLD,RT,ATRM,ICEN,MDAT,HAZ
      COMMON /RTCDAT/ OP,RST,HLD,RT,ATRM,ICEN,MDAT,HAZ
      EQUIVALENCE (FSIM( 1),T     ),(FSIM(12),XLAT),(FSIM(13),XLNG)
C
      DATA R2D /57.29578/
CCC      DATA FIRST /.TRUE./
C
C.... FIRST TIME IN REAL-TIME LOOP, INITIALIZE RANDOM NUMBER
C     GENERATOR.  REQUIRED BECAUSE 'SETUP', WHICH INITIALIZES
C     'RAND' IN THE STANDARD GRAM PROGRAM, IS NOT IN THE REAL-
C     TIME LOAD MODULE IN THE DRYDEN SIM.  LJS.
C
      IF(FIRST) THEN
         RDUM=RAND(1)
         RDUM=RAND(0)
         RDUM=RAND(0)
         FIRST=.FALSE.
         RST=.TRUE.
         OP=.FALSE.
         HLD=.FALSE.
      ELSE
         RST=.FALSE.
         OP=.TRUE.
         HLD=.FALSE.
      ENDIF
C
      GLATF = GLATX / R2D
C
C.... SCALE INPUTS FROM SIM.
C
      IET  = T
      H    = HSIM/3280.84
      PHI  = XLAT*R2D
      THET = XLNG*R2D
      IF(THET.LT.0.0) THET = THET+360.0
C
C.... IF HOLD MODE, RETURN.
C
      IF(HLD) RETURN
C
C.... IF RESET MODE, EXECUTE GRAM FIRST PASS CODE FROM GRAM MAIN
C     PROGRAM.
C
      IF(.NOT.RST) GO TO 200
C
      NT   = 1
      PHIR =PHI /R2D
      THETR=THET/R2D
C
      PHI1R  = PHIR
      THET1R = THETR
C
C.... A=EQUATORIAL EARTH RADIUS, B = POLAR EARTH RADIUS
```

```
C     EPS= EARTH ECCENTRICITY
C
      A  = 6378.160
      B  = 6356.7747
      EPS=(1.-(B*B)/(A*A))
C
C.... COMPUTES RADIUS TO HEIGHT H, AND GRAVITY AT HEIGHT AND
C     LATITUDE PHIR
C
      CALL RIG
      ISEC=ISECO+IET
      ISEC=MOD(ISEC,60)
      MIN = MINO + IET/60
      IHR = IHRO + MIN / 60
      MIN = MOD(MIN,60)
C
C.... COMPUTES P,D,T,U,V AT FIRST POSITION AFTER INITILL POSITION
C
      IF(H.LE.30.) LOOK=1
      IF(ABS(PHIR).GT.GLATF) GO TO 195
      IF(H.GE.25.0 .AND. H.LE.90.0) GO TO 195
      PHI1S=PHI1R
      PHIS=PHIR
      DPHIS=(PHIR+GLATF)/(2.*GLATF)
      PHIR=GLATF
      PHI1R=PHIR+PHI1S-PHIS
C
      CALL SCIMOD(0)
C
      UP2=UPRE
      VP2=VPRE
      DUP2=DUPRE
      DVP2=DVPRE
      PHIR=-GLATF
C
      CALL SCIMOD(0)
C
      UP1=UPRE
      VP1=VPRE
      DUP1=DUPRE
      DVP1=DVPRE
      UPRE=UP1+(UP2-UP1)*DPHIS
      VPRE=VP1+(VP2-VP1)*DPHIS
      DUPRE=DUP1+(DUP2-DUP1)*DPHIS
      DVPRE=DVP1+(DVP2-DVP1)*DPHIS
      PHIR=PHIS
      PHI1R=PHI1S
C
  195 CALL SCIMOD(1)
C
  200 CONTINUE
C
C.... IF OPERATE MODE, CYCLE GRAM PROGRAM.  CODE FROM GRAM 'MAIN'.
C
      IF(.NOT.OP) GO TO 300
C
      NT = NT + 1
      PHIR =PHI /R2D
      THETR=THET/R2D
      MIN=MINO+IET/60
      ISEC=IET
      ISEC=MOD(ISEC,60)
      IHR=IHRO+MIN/60
      MIN=MOD(MIN,60)
```

```
C
C.... COMPUTE RADIUS AND GRAVITY AT NEW POSITION
C
      CALL RIG
C
C.... COMPUTE P,D,T,U,V, AT NEW POSITION
C
      IF(ABS(PHIR).GT.GLATF) GO TO 80
      IF(H.GE.25.0 .AND. H.LE.90.0) GO TO 80
      PHI1S=PHI1R
      PHIS=PHIR
      DPHIS=(PHIR+GLATF)/(2.*GLATF)
      PHIR=GLATF
      PHI1R=PHIR+PHI1S-PHIS
C
      CALL SCIMOD(0)
C
      UP2=UPRE
      VP2=VPRE
      DUP2=DUPRE
      DVP2=DVPRE
      PHIR=-GLATF
C
      CALL SCIMOD(0)
C
      UP1=UPRE
      VP1=VPRE
      DUP1=DUPRE
      DVP1=DVPRE
      UPRE=UP1+(UP2-UP1)*DPHIS
      VPRE=VP1+(VP2-VP1)*DPHIS
      DUPRE=DUP1+(DUP2-DUP1)*DPHIS
      DVPRE=DVP1+(DVP2-DVP1)*DPHIS
      PHIR=PHIS
      PHI1R=PHI1S
C
   80 CALL SCIMOD(1)
C
  300 CONTINUE
C
C.... SCALE GRAM OUTPUTS FOR SIMULATION USE.
C
C.... COMPUTE SPEED OF SOUND IN M/SEC.
C
      CS76MS=SQRT(401.8743*TS )
      CSUNPR=SQRT(401.8743*TGH)
      CSPERT=SQRT(401.8743*TH )
C
C.... CONVERT FROM SI TO ENGLISH UNITS.  '76' SUFFIX USED FOR THE 76
C     STANDARD ATMOSPHERE VALUE.  'U' SUFFIX USE FOR THE GRAM
C     UNPERTURBED (MONTHLY MEAN) VALUE.  'P' SUFFIX USED FOR THE GRAM
C     MEAN PLUS PERTURBATIONS.
C
      CS76  =CS76MS/0.3048          !SPEED OF SOUND IN FT/SEC
      CSU   =CSUNPR/0.3048
      CSP   =CSPERT/0.3048
C
      TMPR76=TS *1.8                !TEMPERATURE IN DEG RANKINE
      TMPRU =TGH*1.8
      TMPRP =TH *1.8
C
      PA76  =PS *0.02088543         !PRESSURE IN LBS/FT**2
      PAU   =PGH*0.02088543
      PAP   =PH *0.02088543
```

```
C
      RHO76 =DS *0.001940319         !DENSITY IN SLUGS/FT**3
      RHOU  =DGH*0.001940319
      RHOP  =DH *0.001940319
C
      UWINDU=VGH/0.3048              !GEOSTROPHIC & TOTAL WINDS FT/SEC
      UWINDP=VH /0.3048              !U,V COMPONENTS INTERCHANGE
      VWINDU=UGH/0.3048              !BETWEEN GRAM AND SIM AXES.
      VWINDP=UH /0.3048
C
      USHEAR=DVH/1000.0              !WIND SHEAR (FT/SEC)/FT
      VSHEAR=DUH/1000.0
C
C.... SELECT OUTPUT TO SIM BASED ON USER SELECTION.  IT IS POSSIBLE
C     FOR THE USER TO SELECT GRAM ATMOSPHERE AND GRAM WINDS
C     INDEPENDENTLY.
C
      IF(.NOT.GRMATM) GO TO 50
C
      IF(G76ATM) THEN          ! GRAM 76 REFERENCE SELECTED
         ASIM = CS76
         RHO  = RHO76
         PA   = PA76
         TMPR = TMPR76
      ELSE                     ! GRAM CALCULATED ATMOSPHERE SELECTED
         IF(GATMP ) THEN       ! GRAM MONTHLY MEAN + PERTURBATIONS
            ASIM = CSP
            RHO  = RHOP
            PA   = PAP
            TMPR = TMPRP
         ELSE                  ! GRAM MONTHLY MEAN WITHOUT PERT.
            ASIM = CSU
            RHO  = RHOU
            PA   = PAU
            TMPR = TMPRU
         ENDIF
      ENDIF
C
   50 CONTINUE
C
C.... TEST FOR USER SELECTION OF WINDS.
C
      IF(.NOT.GRMWND) GO TO 60
C
      IF(WIND) THEN              ! GRAM WINDS SELECTED, WIND ON
         IF(GWINDP) THEN         ! MEAN + PERTURBATIONS SELECTED
            XWIND = UWINDP
            YWIND = VWINDP
         ELSE                    ! MONTHLY MEAN WITHOUT PERT.
            XWIND = UWINDU
            YWIND = VWINDU
         ENDIF
      ELSE                       ! GRAM WINDS SELECTED, BUT WINDS OFF.
         XWIND = 0.0
         YWIND = 0.0
      ENDIF
C
   60 CONTINUE
C
      RETURN
      END
C
      SUBROUTINE GRIDIN
C
```

```
C.... THIS ROUTINE READS A BINARY DATA FILE CONSISTING OF PRESSURE,
C     DENSITY, TEMPERATURE, PRESSURE VARIANCE, DENSITY VARIANCE, AND
C     TEMPERATURE VARIANCE FOR LATITUDES 20-65 AND LONGITUDES 35-140
C     WEST (CONTINENTAL US +).  THE DATA TABLE IS USED TO SUPPLY THE
C     REAL-TIME GRAM PROGRAM WITH 4D GRID DATA WHEN WITHIN THE REGION
C     INDICATED ABOVE (CONUS INCLUDING A SORROUNDING AREA).  THE
C     DATA IS USED BY ROUTINE 'USGRID' TO BUILD A GRID.  'USGRID'
C     REPLACES 'GEN4D', 'ADJUST', 'GRID4D', 'INTRP4', 'SELEC4', AND
C     'SORT4'.
C
C.... THIS ROUTINE REQUIRES THE INCLUSION OF THE EXTENDED REGION.  THE
C     REGION IS CREATED ON THE SYSTEM VOLUME WITH THE FOLLOWING
C     VOLUME MANAGER COMMAND:
C        CREATE COMMON NASPGM FIRST=380 PROT=68 ACCESS=OT(R W)
C
C.... ROUTINE I/O:    LUN 25 IS INPUT FILE
C                     'UT'  IS TERMINAL OUTPUT
C
C.... WRITTEN   23 JAN 89   L SCHILLING   NASA/ADFRF.
C
      CHARACTER*12 FILNAM
      COMMON /PDTCOM/ IT,MONTH,DUMMY1(8118)
      COMMON /NASPGM/ PDAT(5720) ,DDAT(5720) ,TDAT(5720) ,
     .                          SPDAT(5720),SDDAT(5720),STDAT(5720)
C
C.... OPEN BINARY FILE CONTAINING US 4D GRID DATA FOR MONTH OF
C     INTEREST.
C
      WRITE(FILNAM,777) MONTH
  777 FORMAT('NASPGRID',I2,'.B')
      IF(FILNAM(9:9).EQ.' ') FILNAM(9:9)='0'
C
      OPEN(25,FILE=FILNAM,STATUS='OLD',FORM='UNFORMATTED',
     .     ACCESS='SEQUENTIAL',ERR=99,IOSTAT=IOS)
C
C.... READ IN BINARY DATA FILE.
C
      REWIND(25)
      READ(25) PDAT,TDAT,DDAT,SPDAT,STDAT,SDDAT
C
      CLOSE(25)
C
      RETURN
C
C.... ERROR CONDITION.
C
   99 CONTINUE
C
      WRITE(6,231) FILNAM,IOS
  231 FORMAT('  OPEN ERROR ON ',A12,'  STATUS=',I3)
C
      STOP
      END
C
      SUBROUTINE GTEPP(IH,PHI,P,D,T,PG,DG,TG,DPY,DTY,DP2Y)
C
C.....INTERPOLATES GROVES DATA TO HEIGHT IH AND LATITUDE PHI
C
C.... DECLARE APPROPRIATE ARGUMENTS TO BE IN GOULD EXTENDED.  IF A
C     NON-EXTENDED PARAMETER IS DECLARED EXTENDED, THE INTERFACE WILL
C     WORK, ALBEIT WITH A SLIGHT EXECUTION OVERHEAD PENALTY.  LJS.
C
C
      DIMENSION PG(18,19),TG(18,19),DG(18,19)
```

```
C      HEIGHT INDEX
       I =(IH - 20)/5
C      LOWER LATITUDE INDEX
       J = INT((PHI + 100.)/10.)
       IF (J.LT.1) J = 1
       IF (J.GT.18) J = 18
C      UPPER LATITUDE INDEX
       JP = J + 1
C.....CHECK FOR DENSITY OR TEMPERATURE LEQ 0
       CHK = DG(I,J) * TG(I,J) * DG(I,JP) * TG(I,JP)
       IF (CHK) 10,10,20
 10    P = PG(I,J)
       D = DG(I,J)
       T = TG(I,J)
       GO TO 30
C.....LATITUDE DEVIATION FROM GROVES ARRAY POSITION
 20    PHIF = (PHI + 100. - 10.*J)/10.
       TL= TG(I,J) + (TG(I,JP ) - TG(I,J))*PHIF
C       LATITUDE INTERPOLATION
       DL= DG(I,J) + (DG(I,JP ) - DG(I,J)) *PHIF
       R1 = PG(I,J)/(DG(I,J)*TG(I,J))
       R2 = PG(I,JP)/(DG(I,JP)*TG(I,JP))
C      INTERPOLATED GAS CONSTANT
       R = R1 + (R2 - R1)*PHIF
C      PRESSURE COMPUTED FROM INTERPOLATED GAS CONSTANT
       P =DL*R*TL
       D = DL
       T = TL
C      DP/DY FOR GEOSTOPHIC WINDS
 30    DPY = (PG(I,JP) - PG(I,J)) * 0.5
C      DT/DY FOR THERMAL WINDS
       DTY = (TG(I,JP) - TG(I,J)) * 0.5
       JM = J - 1
       IF (JM.LT.1) JM = JP
       DP2Y = (PG(I,JP) - PG(I,JM ))*0.5
       IF (ABS(PHI)-90.) 50,40,40
40     DPY = 0.
       DTY = 0.
       DP2Y = 0.
50     CONTINUE
       RETURN
       END
C
       SUBROUTINE INTLL(F,IA,IB,IC,ID,FLL,GLAT,GLON,CLAT,CLON,IH)
C
C.....INTERPOLATES FUNCTION (ARRAY) F FROM VALUES OF GLAT AND GLON AT
C         INDEX VALUES IA, IB, IC, ID TO OUTPUT VALUE FLL AT HEIGHT IH
C         AND POSITION CLAT, CLON
C
       DIMENSION F(16,26),GLAT(16),GLON(16)
C.....NORMALIZES LONGITUDE DISPLACEMENT
       IF(F(IA,IH)*F(IB,IH)*F(IC,IH)*F(ID,IH)) 20,10,20
    10 FLL=0.
       RETURN
    20 X=(CLON-GLON(IB))/(GLON(IA)-GLON(IB))
C.....NORMALIZES LATITUDE DISPLACEMENT
       Y=(CLAT-GLAT(IA))/(GLAT(IC)-GLAT(IA))
C.....TWO DIMENSIONAL INTERPOLATION
       FLL=F(IB,IH)+(F(ID,IH)-F(IB,IH))*Y+(F(IA,IH)-F(IB,IH))*X
     1   +(F(IC,IH)-F(IA,IH)-F(ID,IH)+F(IB,IH))*X*Y
       RETURN
       END
       SUBROUTINE INTRUV(UR,VR,H,PHI,SUH,SVH)
C
```

```
C.....FINDS RANDOM WIND STANDARD DEVIATION AT HEIGHT H (KM), LATITUDE
C         PHI (DEGREES), FROM UR AND VR ARRAYS
C
      DIMENSION UR(25,10),VR(25,10)
C.....I - LOWER HEIGHT INDEX
      IF (H.LT.95.) I = 1 + INT(H) / 5
      IF (H.GE.95.) I=19+(INT(H)-80)/20
      IF (I.GT.25) I = 25
C     UPPER HEIGHT INDEX
      IP=I+1
      IF (IP.GT.25) IP=25
C     LOWER LATITUDE INDEX
      J=INT(PHI+110.)/20
C      UPPER LATITUDE INDEX
      JP=J+1
      IF (JP.GT.10) JP=10
C.....PHI1 - LOWER LATITUDE FOR UR AND VR ARRAY VALUES
      PHI1=-110.+20.*J
C.....PHI2 - UPPER LATITUDE FOR UR AND VR ARRAY VALUES
      PHI2=-110.+20.*JP
      IF (I.GT.19) GO TO 10
C      LOWER HEIGHT FOR UR AND VR ARRAY VALUES
      Z1=5.*(I-1)
      GO TO 20
10    Z1=20.*(I-15)
20    IF (IP.GT.19) GO TO 30
C      UPPER HEIGHT FOR UR AND VR ARRAY VALUES
      Z2=5.*(IP-1)
      GO TO 40
   30 Z2 = 20. * (IP - 15)
C     INTERPOLATE ON LATITUDE AT LOWER HEIGHT
   40 CALL INTERW(UR(I,J),VR(I,J),PHI1,UR(I,JP),VR(I,JP),PHI2,U1,V1,
     $ PHI)
C      INTERPOLATE ON LATITUDE AT UPPER HEIGHT
      CALL INTERW(UR(IP,J),VR(IP,J),PHI1,UR(IP,JP),VR(IP,JP),PHI2,U2,
     $ V2,PHI)
C      INTERPOLATE ON HEIGHT
      CALL INTERW(U1,V1,Z1,U2,V2,Z2,SUH,SVH,H)
      RETURN
      END
      SUBROUTINE INTERW(U1,V1,Z1,U2,V2,Z2,U,V,Z)
C
      IF ( Z1 - Z2 ) 20,10,20
   10 U = U1
C     SETS U,V = U1,V1 IF Z1 = Z2
      V = V1
      RETURN
   20 A = (Z-Z1)/(Z2-Z1)
      U = U1 + (U2-U1) * A
      V = V1 + (V2-V1) * A
C.....LINEAR INTERPOLATION BETWEEN U1,V1 AT HEIGHT Z1 AND U2,V2 AT
C         HEIGHT Z2.  OUTPUT IS U,V AT HEIGHT Z
      RETURN
      END
      SUBROUTINE INTERZ(P1,D1,T1,Z1,P2,D2,T2,Z2,P,D,T,Z)
C
    5 IF (Z1 - Z2)  20.10,20
 10   P = P1
      D = D1
C     SETS P, D, T = P1.D1,T1, IF Z1 = Z2
      T =T1
      RETURN
 20   A = (Z - Z1)/ (Z2 - Z1)
      T = T1 + (T2 - T1) * A
      D = D1 + (D2 - D1) * A
      P = P1 + (P2 - P1) * A
C.....LINEAR INTERPOLATION BETWEEN P1,D1,T1 AT HEIGHT Z1 AND P2,D2,T2
C         AT HEIGHT Z2 TO OUTPUT VALUES OF P,D,T AT HEIGHT Z
      RETURN
      END
      SUBROUTINE INTER2(P1,D1,T1,Z1,P2,D2,T2,Z2,P,D,T,Z)
C.....INTERPOLATES BETWEEN P1,D1,T1 AT HEIGHT Z1 AND P2,D2,T2 AT
C         HEIGHT Z2 TO OUTPUT VALUES OF P,D,T AT HEIGHT Z
C.....CHECKS FOR T1,D1,T2,D2 PRODUCT = 0, FOR GAS CONSTANT INTERPOLATION
      CHK=T1*D1*T2*D2
      IF (CHK) 10,10,5
    5 IF (Z1 - Z2)  20,10,20
 10   P = P1
      D = D1
C       SETS P,D,T = P1,D1,T1 IF Z1=Z2
      T = T1
      RETURN
   20 IF(P1*D1*T1*P2*D2*T2.LE.0.)GO TO 30
      IF(D2*D1.LE.0.0)GO TO 30
      A=ALOG(D2/D1)/(Z2-Z1)
C      LINEAR INTERPOLATION ON LOG D
      DZ= D1*EXP(A*(Z - Z1))
      A=(Z-Z1)/(Z2-Z1)
C      LINEAR INTERPOLATION ON T
      TZ= T1 + A*(T2-T1)
      R1=P1/(D1*T1)
      R2=P2/(D2*T2)
C      LINEAR INTERPOLATION ON GAS CONSTANT R
      R=(R2-R1)*A+R1
C       PRESSURE FROM PERFECT GAS LAW
      P = DZ * R * TZ
      D = DZ
      T = TZ
      RETURN
   30 P=0.
      D=0.
      T=0.
      RETURN
      END
      SUBROUTINE INTER4 (            CLAT, CLON, IZ,     P, D, T,
     $ P4, D4, T4, DPX, DPY, DTX, DTY,DPXX,DPYY,DPXY)
C
      COMMON/IOTEMP/IOTEM1,IOTEM2,IUG, IUN ,DD,XMJD,PHI1,PHI,
     $NSAME,DUMMY2(56)
C.....INTERPOLATES BETWEEN 4D ARRAYS P(I,IH),D(I,IH),T(I,IH) AT GRID
C         LOCATIONS LATITUDE GLAT(I) LONGITUDE GLON(I).
C         CLAT,CLON = CURRENT LATITUDE,LONGITUDE
C         IZ = HEIGHT                     NG = NUMBER OF 4D GRID POSITIONS
C         OUTPUT = P4,D4,T4, AND DERIVATIVES DPX,DPY,DTX,DTY
      COMMON /C4    / GLAT(16),GLON(16),NG,DUMMY(2499)
      COMMON/CHIC/LA(4,4),NB(2),IWSYM,UCOEF(14,9),VCOEF(14,9)
      DIMENSION                   P(16,26),D(16,26),T(16,26),LAX(16)
      DATA IBLK/1H /,IAST/1H*/
      IWSYM = IBLK
      ICHK = 0
C     HEIGHT INDEX = HEIGHT + 1
      IH = IZ + 1
  5   IF (ICHK.GT.1) GO TO 220
      IF (NG.GT.9) GO TO 100
C     NG = 9 MEANS POLAR GRID
      DO 10 I=10,16,1
      P(I,IH) = P(9,IH)
      D(I,IH) = D(9,IH)
      T(I,IH) = T(9,IH)
      GLAT(I) = GLAT(9)
C     I=10-16 ALL AT 90 DEG
   10 GLON(I) = GLON(I-8)
C     LOWER RIGHT INTERPOLATION INDEX
      IB = INT(CLON/45) + 1
C     LOWER LEFT INTERPOLATION INDEX
      IA =IB+1
      IF (IA.GT.8) IA = IA-8
C     POSITION OUTSIDE POLAR GRID
      IF (ABS(CLAT).LT.75.) GO TO 20
C     UPPER LEFT INTERPOLATION INDEX
      IC = IA +8
C     UPPER RIGHT INTERPOLATION INDEX
      ID = IB + 8
      GO TO 300
 20   IF(NSAME.EQ.1) NSAME=2
      CALL GEN4D
C::_
C      CALL USGRID
C::^
      ICHK = ICHK + 1
      GO TO 5
  100 XLON = CLON
      DO 105 I = 1,4
      DO 105 J = 1,4
      I16 = 4*(I-1) + J
      LAX(I16) = LA(I,J)
 105   CONTINUE
      IF (XLON-GLON(1).GT.180)XLON=CLON-360.
C.....CHECKS FOR POSITION WITHIN 16 POINT GRID 110=GOOD.  200=POSITION
C         OUTSIDE GRID.
      IF (CLAT.GE.GLAT(1) .AND. CLAT.LT.GLAT(16) .AND. XLON.LE.GLON(1)
     $  .AND.XLON.GT.GLON(16)) GO TO 110
      GO TO 200
 110  NDL=5
      IF(ABS(CLAT).LT.18)    =12
      IA = 1 + INT((GLON(1)  - XLON) / 5)
C.....IA = LOWER LEFT (REFERENCE) INTERPOLATION INDEX
      IA = IA + 4 * INT((CLAT - GLAT(1)) / NDL)
C     LOWER RIGHT INTERPOLATION INDEX
      IB = IA + 1
C     UPPER LEFT INTERPOLATION INDEX
      IC = IA + 4
C     UPPER RIGHT INTERPOLATION INDEX
      ID = IA + 5
      GO TO 300
 200  IF(NSAME.EQ.1)NSAME=2
      CALL GEN4D
C::_
C      CALL USGRID
C::^
      ICHK = ICHK + 1
      GO TO 5
  220 CONTINUE
C::  220 WRITE(6,250)
C::  250 FORMAT(1H ,'UNABLE TO GENERATE 4-D GRID.  TOO MANY ',
C::      &'RETRIES IN INTER4')
      P4=0.
      D4=0.
      T4=0.
      RETURN
C.....INTERPOLATION FOR POSITION INSIDE 16 POINT GRID OR POLAR GRID
  300 CALL INTLL(P,IA,IB,IC,ID,P4,GLAT,GLON,CLAT,XLON,IH)
```

```
      CALL INTLL(D,IA,IB,IC,ID,D4,GLAT,GLON,CLAT,XLON,IH)
      CALL INTLL(T,IA,IB,IC,ID,T4,GLAT,GLON,CLAT,XLON,IH)
C.....RELATIVE LONGITUDE DISPLACEMENT FROM REFERENCE POSITION (IA)
      DLON = (XLON - GLON(IA))/(GLON(IB) - GLON(IA))
C.....RELATIVE LATITUDE DISPLACEMENT FROM REFERENCE POSITION(IA)
      DLAT = (CLAT - GLAT(IA))/(GLAT(IC) - GLAT(IA))
      DPX=P(IB,IH)-P(IA,IH)
C.....DP/DX FOR GEOSTROPHIC WIND EQUATIONS
      DPX = DPX + (P(ID,IH) - P(IC,IH) - DPX)*DLAT
      DTX = T(IB,IH) - T(IA,IH)
C.....DT/DX FOR THERMAL WIND EQUATIONS
      DTX = DTX + (T(ID,IH) - T(IC,IH) - DTX)*DLAT
      DPY = P(IC,IH) - P(IA,IH)
C.....DP/DY FOR GEOSTROPHIC WIND EQUATIONS
      DPY = DPY + (P(ID,IH) - P(IB,IH) - DPY)*DLON
      DTY = T(IC,IH) - T(IA,IH)
C.....DT/DY FOR THERMAL WIND EQUATIONS
      DTY = DTY + (T(ID,IH) - T(IB,IH) - DTY)*DLON
      IF(NG.GT.9) GO TO 315
      DPX=DPX/9.
      DTX=DTX/9.
      DPY=DPY/3.
      DTY=DTY/3.
  315 IF(ABS(CLAT).GT.18) GO TO 312
      DPY=DPY*5./12
      DTY=DTY*5./12
  312 IF (NG.GT.9) GO TO 310
      DPXX = 0.
      DPYY = 0.
      DPXY = 0.
      RETURN
  310 DPXY = P(ID,IH) - P(IC,IH) - P(IB,IH) + P(IA,IH)
      IF (MOD(IB,4) .EQ.0) GO TO 320
      I1 = IA
      I2 = IB + 1
      I3 = IC
      I4 = ID + 1
      SX=1.
      GO TO 330
  320 I1 = IA - 1
      I2 = IB
      I3 = IC - 1
      I4 = ID
      SX=-1.
  330 IF(LAX(I1).NE.LAX(IA).OR.LAX(I2).NE.LAX(IA).OR.LAX(I3).NE.
     * LAX(IA).OR.LAX(I4).NE.LAX(IA)) GO TO 360
      DPXX = P(I2,IH) - P(I1,IH)
      DPXX = DPXX + (P(I4,IH) - P(I3,IH) - DPXX)*DLAT
      IF (IC.GT.12) GO TO 340
      I1 = IA
      I2 = IC + 4
      I3 = IB
      I4 = ID + 4
      SY=1.
      GO TO 350
  340 I1 = IA - 4
      I2 = IC
      I3 = IB - 4
      I4 = ID
      SY=-1.
  350 IF(LAX(I1).NE.LAX(IA).OR.LAX(I2).NE.LAX(IA).OR.LAX(I3).NE.
     * LAX(IA).OR.LAX(I4).NE.LAX(IA)) GO TO 360
      DPYY = P(I2,IH) - P(I1,IH)
      DPYY = DPYY + (P(I4,IH) - P(I3,IH) - DPYY)*DLON
      DPXX =(DPXX - 2.*DPX )*SX
      DPYY =(DPYY - 2.*DPY )*SY
      RETURN
 360  DPXX = 0.
      DPYY = 0.
      DPXY = 0.
      RETURN
      END
      SUBROUTINE JAC(Z,TZ,DENS)
      COMMON/IOTEMP/IOTEM1,IOTEM2,IUG,IUN,DD,XMJD,PHI1,PHI,
     .           NSAME,RP1, RD1, RT1, SP1, SD1, ST1, RU1, RV1, SU1, SV1,
     $ MN, IDA, IYR, H1, PHI1R,THET1R,G,RI,H,PHIR,THETR,F10,F10B,AP,
     $ IHR,MIN,NMORE,DX,HL,VL,DZ,DUMMY(25)
      COMMON/COMJAC/XLAT,XLONG,SDA,SHA,DY,Y,T,EM
      DIMENSION ALPHA(6),EI(6),DI(6),      B(7),DIT(6)
      QQ = 100.
      DATA ALPHA/0.0,0.0,0.0,0.0,-0.38,0.0/
      DATA EI/28.0134,31.9988,15.9994,39.948,4.0026,1.00797/
      DATA B/28.15204,-0.085586,1.284E-04,-1.0056E-05,-1.021E-05,
     11.5044E-06,9.9826E-08/
      AV=6.02257E23
      QN=.78110
      QO2=.20955
      QA=.009343
      QHE = 1.289E-5
      FK=8.31432
C
C     TEMPERATURE AT Z = 125 KM, EQ. 9
C
      TX=444.3807+.02385*T -392.8292*EXP(-.0021357*T)
      A2=2.*(T-TX)/3.14159265
C
C
      DIT(6)=0.
      M=10
      EPS=.0001
C
C     TEMPERATURE FOR 90%Z%125, EQ. 10
C
      T1=1.9*(TX-183.)/35.
      T4=3.*(TX-183.-2.*T1*35./3.)/(35.**4)
      T3=-T1/(3.*35.**2)+4.*T4*35./3.
      TZ=TX+T1*(Z-125.)+T3*(Z-125.)**3+T4*(Z-125.)**4
      IF (Z-105.) 43,43,40
C
C     MEAN MOLECULAR WEIGHT FOR 90%Z%105, EQ. 1
C
   43 Z2 = Z - QQ
      EM=B(1)+B(2)*Z2+B(3)*Z2**2+B(4)*Z2**3+B(5)*Z2**4+B(6)*Z2**5
     1+B(7)*Z2**6
      D=Z
70    CONTINUE
C
C     INTEGRATION OF EQ. 5 FOR DENSITY BETWEEN 90%Z%105
C
      A=90.
      FA=B(1)+B(2)*(A-QQ)+B(3)*(A-QQ)**2+B(4)*(A-QQ)**3+B(5)*(A-QQ)**4
     1+B(6)*(A-QQ)**5 +B(7)*(A-QQ)**6
      FA=FA*9.80655/((1.+A/6.356766E+3)**2)
      FA=FA/(TX+T1*(A-125.)+T3*(A-125.)**3 +T4*(A-125.)**4)
      FD=B(1)+B(2)*(D-QQ)+B(3)*(D-QQ)**2+B(4)*(D-QQ)**3+B(5)*(D-QQ)**4
     1+B(6)*(D-QQ)**5 +B(7)*(D-QQ)**6
      FD=FD*9.80665/((1.+D/6.356766E+3)**2)
      FD=FD/(TX+T1*(D-125.)+T3*(D-125.)**3 +T4*(D-125.)**4)
```

```
C      SRQ4, SIMPSONS RULE QUADRATURE  -  G.F.KUNCIR
C      DEFINITIONS  -
C        A = LOWER LIMIT OF INTEGRATION
C        D = UPPER LIMIT OF INTEGRATION
C        FUNC = INTEGRAND FUNCTION SUBPROGRAM
C        EPS = RELATIVE ERROR CONVERGENCE CRITERION
C        M = MAXIMUM NUMBER OF INTEGRATIONS
C        R = RESULT OF INTEGRATION
C        N = NUMBER OF INTEGRATIONS9RIQ&IRID TO FIND R
C
       NINT = 1
       N=0
       PREV=0.
       SONE=(D-A)*(FA+FD)/2.
71     N=N+1
       IF (N-M) 72,72,75
72     NINT = 2 * NINT
       STWO=0.
       DEL=(D-A)/FLOAT(NINT)
       DO 73 I=1,NINT,2
       X=A+DEL*FLOAT(I)
       FX=B(1)+B(2)*(X-QQ)+B(3)*(X-QQ)**2+B(4)*(X-QQ)**3+B(5)*(X-QQ)**4
      1+B(6)*(X-QQ)**5 +B(7)*(X-QQ)**6
       FX=FX*9.80665/((1.+X/6.356766E+3)**2)
       FX=FX/(TX+T1*(X-125.)+T3*(X-125.)**3 +T4*(X-125.)**4)
73     STWO=STWO+FX
       CUR=SONE+4.*DEL*STWO
       IF (EPS*ABS(CUR)-ABS(CUR-PREV)) 74,75,75
74     PREV=CUR
       SONE=(SONE+CUR)/4.
       GO TO 71
75     R=CUR/3
       IF (Z-105.) 44,76,44
   44  IF (D-105.) 76,55,76
C
C      DENSITY FOR 90%Z%105
C
76     DENS=3.46E-9*183.*EM*EXP(-R/FK)/(TZ*28.878)
       DL=ALOG10(DENS)
       PAR=AV*DENS/EM
       AN=ALOG10(QN*EM*PAR/28.96)
       AA=ALOG10(QA*EM*PAR/28.96)
       AHE=ALOG10(QHE*EM*PAR/28.96)
       AO=ALOG10(2.*PAR*(1.-EM/28.96))
       AO2=ALOG10(PAR*(EM*(1.+QO2)/28.96-1.))
       AH=-0.
       RETURN
C
C      TEMPERATURE AND MEAN MOLECULAR WEIGHT AT Z=105 KM
C
     40 Z3=105.
       TZ3=TX+T1*(Z3-125.)+T3*(Z3-125)**3+T4*(Z3-125)**4
       ZM3=B(1)+B(2)* 5.+B(3)* 25.+B(4)* 125.+B(5)* 5.**4.+B(6)* 5.**5.
      1+B(7)* 5.**6.
       D=105.
       GO TO 70
C
C      DENSITY  AT Z=105 KM
C
55     DEN1=3.46E-9*183.*ZM3*EXP(-R/FK)/(TZ3*28.878)
       PAR=AV*DEN1/ZM3
       DI(1)=QN*ZM3*PAR/28.96
       DI(2)=PAR*(ZM3*(1.+QO2)/28.96-1.)
       DI(3)=2.*PAR*(1.-ZM3/28.96)
      DI(4)=QA*ZM3*PAR/28.96
      DI(5)=QHE*ZM3*PAR/28.96
      IF(Z-125.) 56,56,90
 56   CONTINUE
C
C     INTEGRATION OF EQ. 6 FOR DENSITY ABOVE 105 KM
C
      R=0.
      D1=125.
      A1=105.
  400 CONTINUE
      FA1=9.80665/((1.+A1/6.356766E+3)**2)
      FA1=FA1/(TX+T1*(A1-125.)+T3*(A1-125.)**3+T4*(A1-125.)**4)
      FD1=9.80665/((1.+D1/6.356766E+3)**2)
      IF(D1-125.) 45,45,50
   45 FD1=FD1/(TX+T1*(D1-125.)+T3*(D1-125.)**3+T4*(D1-125.)**4)
      GO TO 51
 50   FD1=FD1/(TX+A2*ATAN(T1*(D1-125.)*(1.+4.5E-6*(D1-125.)**2.5)/A2))
      TZ=TX+A2*ATAN(T1*(Z-125.)*(1.+4.5E-6*(Z-125.)**2.5)/A2)
 51   N=0
      NINT = 1
      PREV=0
      SONE=(D1-A1)*(FA1+FD1)/2.
 81   N=N+1
      IF (N-M) 82,82,85
 82   NINT = 2 * NINT
      STWO=0.
      DEL=(D1-A1)/FLOAT(NINT)
      DO 83 I=1,NINT,2
      X1=A1+DEL*FLOAT(I)
      FX1=9.80665/((1.+X1/6.356766E+3)**2)
      IF(X1-125.) 46,46,52
   46 FX1=FX1/(TX+T1*(X1-125.)+T3*(X1-125.)**3+T4*(X1-125.)**4)
      GO TO 83
 52   FX1=FX1/(TX+A2*ATAN(T1*(X1-125.)*(1.+4.5E-6*(X1-125.)**2.5)/A2))
 83   STWO=STWO+FX1
      CUR=SONE+4.*DEL*STWO
      IF (EPS*ABS(CUR)-ABS(CUR-PREV)) 84,85,85
 84   PREV=CUR
      SONE=(SONE+CUR)/4.
      GO TO 81
 85   R=CUR/3.+R
      IF(A1.EQ.125.) GO TO 430
      D1=Z
      A1=125.
      GO TO 400
  430 CONTINUE
C
C     DENSITY ABOVE 105 KM
C
      DO 41  I=1,5
      DIT(I)=DI(I)*(TZ3/TZ)**(1.+ALPHA(I))*EXP(-EI(I)*R/FK)
   41 CONTINUE
      DENS=0
      DO 42 I=1,6
      DENS=DENS+EI(I)*DIT(I)/AV
 42   CONTINUE
C
C     MEAN MOLECULAR WEIGHT FOR Z 105 KM
C
      EM=DENS*AV/(DIT(1)+DIT(2)+DIT(3)+DIT(4)+DIT(5)+DIT(6))
C
C     LOG DENSITY
C
```

```
      DL=ALOG10(DENS)
      AN =ALOG10(DIT(1))
      AO2=ALOG10(DIT(2))
      AO =ALOG10(DIT(3))
      AA =ALOG10(DIT(4))
      AHE=ALOG10(DIT(5))
      IF(Z-500.) 47,48,48
   47 DIT(6)=10.**(-6)
   48 AH=ALOG10(DIT(6))
      AN =AMAX1(-0., AN)
      AO2=AMAX1(-0.,AO2)
      AO =AMAX1(-0., AO)
      AA =AMAX1(-0., AA)
      AHE=AMAX1(-0.,AHE)
      AH =AMAX1(-0., AH)
      RETURN
C
C     TEMPERATURE AND DENSITY AT Z=500 KM
C
 90   S=TX+A2*ATAN(T1*375.*(1.+4.5E-6*375.**2.5)/A2)
      DI(6)=10.**(73.13-39.4*ALOG10(S)+5.5*ALOG10(S)*ALOG10(S))
      A1=500.
      IF(Z-500.) 49,60,60
C
C     INTEGRATION OF EQ. 6 FOR DENSITY FOR Z 125 KM
C
   49 A1=Z
   60 FA1=9.80665/((1.+A1/6.356766E+3)**2)
      FA1=FA1/(TX+A2*ATAN(T1*(A1-125.)*(1.+4.5E-6*(A1-125.)**2.5)/A2))
      D1=Z
      IF(Z-500.) 61,62,62
   61 D1=500.
   62 FD1=9.80665/((1.+D1/6.356766E+3)**2)
      FD1=FD1/(TX+A2*ATAN(T1*(D1-125.)*(1.+4.5E-6*(D1-125.)**2.5)/A2))
      N=0
      NINT = 1
      PREV=0
      SONE=(D1-A1)*(FA1+FD1)/2.
 91   N=N+1
      IF (N-M) 92,92,95
 92   NINT = 2 * NINT
      STWO=0.
      DEL=(D1-A1)/FLOAT(NINT)
      DO 93 I=1,NINT,2
      X1=A1+DEL*FLOAT(I)
      FX1=9.80665/((1.+X1/6.356766E+3)**2)
      FX1=FX1/(TX+A2*ATAN(T1*(X1-125.)*(1.+4.5E-6*(X1-125.)**2.5)/A2))
 93   STWO=STWO+FX1
      CUR=SONE+4.*DEL*STWO
      IF (EPS*ABS(CUR)-ABS(CUR-PREV)) 94,95,95
 94   PREV=CUR
      SONE=(SONE+CUR)/4.
      GO TO 91
 95   R=CUR/3.
C
C     TEMPERATURE AT Z 500 KM
C
      TZ=TX+A2*ATAN(T1*(Z-125.)*(1.+4.5E-6*(Z-125.)**2.5)/A2)
      IF(Z-500.) 63,64,64
   63 R=-R
C
C     DENSITY OF HYDROGEN FOR Z 500 KM
C
   64 DIT(6)=DI(6)*(S/TZ)*EXP(-EI(6)*R/FK)
```

```
      GO TO 56
      END
      SUBROUTINE JACCH(Z,PHIR,THET,PH,DH,TH)
C
C.... DECLARE APPROPRIATE ARGUMENTS TO BE IN GOULD EXTENDED.  IF A
C     NON-EXTENDED PARAMETER IS DECLARED EXTENDED, THE INTERFACE WILL
C     WORK, ALBEIT WITH A SLIGHT EXECUTION OVERHEAD PENALTY.  LJS.
C
      COMMON/COMJAC/XLAT,XLONG,SDA,SHA,DY,R,T,EM
      COMMON/IOTEMP/IOTEM1,IOTEM2,IUG,IUN,DD,XMJD,PHI1,PHI,
     .           NSAME,RP1, RD1, RT1, SP1, SD1, ST1, RU1, RV1, SU1, SV1,
     $ M , IDA, IYR, H1, PHI1R,THET1R,G,RI,H,CLAT,CLON ,F10,F10B,AP,
     $ IHR,MIN,NMORE,DX,HL,VL,DZ,DUMMY(25)
C
C     JACCH CALCULATES THE PRESSURE, DENSITY, AND TEMPERATURE AT A
C     POINT IN SPACE ABOVE 90 KM FOR A PARTICULAR TIME
C
C     INPUT
C     Z = HEIGHT IN KM
C     PHIR = LATITUDE IN RADIANS
C     THET = LONGITUDE IN DEGREES  (0 TO 360 DEGREES TURNING WESTWARD)
C     F10 = SOLAR RADIO NOISE FLUX (XE - 22 WATTS/M**2)
C     F10B = 81-DAY AVERAGE F10
C     AP = GEOMAGNETIC INDEX
C     M  = MONTH (FOR YEARLY MEAN VARIABLES M IS SET TO 13)
C     IDA = DAY OF MONTH
C     IYR = YEAR
C     IHR = HOUR OF DAY  (UNIVERSAL TIME)
C     MIN = MINUTE  (UNIVERSAL TIME)
C     XMJD = MEAN JULIAN DAY  (SET EQUAL TO ZERO FOR ANNUAL MEAN)
C     DD = DAY NUMBER WITH RESPECT TO JAN 0 OF YEAR IYR
C
C     OUTPUT
C     PH = PRESSURE IN UNITS OF NT/M**2
C     DH = DENSITY IN UNITS OF KG/M**3
C     TH = TEMPERATURE IN KELVIN DEGREES
C
C     DD = DAY NUMBER WITH RESPECT TO JAN 1 OF YEAR IYR
C
C     REPLACEMENT OF SUBROUTINE VARIABLES TO INSURE NO CHANGES IN THEM
C
      R = 0.31
      XLAT = PHIR
      XLONG = THET
      IF (M.EQ.13) GO TO 50
C
C     CALCULATE SOLAR DEC. AND HOUR ANGLE
C
      CALL TME
C
C     EXOSPHERIC TEMPERATURE
C
      CALL TINF
      GO TO 75
   50 T = 1000.0
C
C     TEMPERATURE, MOLECULAR WEIGHT, AND DENSITY WITHOUT SEASONAL
C        VARIATIONS
C
   75 CALL JAC(Z,TH,DH)
      IF (M.EQ.13) GO TO 300
      YDA = 365.0
      J1 = MOD(IYR,4)
      IF (J1.EQ.0) YDA = 366.0
      C1 = SIN((360. / YDA) * 0.0174532925 * (DD + 100.0))
      IF (PHIR) 80,70,80
   70 C2 = 0.0
      GO TO 90
   80 C2 = (SIN(PHIR) ** 2) * (PHIR / ABS(PHIR))
C
C     DENSITY WITH SEASONAL VARIATIONS
C
   90 Z90 = Z - 90.0
      DLRHO = 0.02 * Z90 * EXP(-0.045 * Z90) * C1 * C2
      DH = DH * EXP(DLRHO)
C
C     MOLECULAR WEIGHT WITH SEASONAL VARIATION
C
      IF (Z - 120.0) 100,100,150
  100 EM = EM + 0.006 * Z90 * C1
      GO TO 250
  150 IF (Z - 230.0) 200,250,250
  200 DEM = EXP(-0.02424 * Z90) * (0.0316 * Z90 - 0.0002257 * Z90 * Z90)
      EM = EM + DEM * C1*0.5
C
C     TEMPERATURE WITH SEASONAL VARIATIONS
C
  250 IF (Z-260.0) 270,300,300
  270 Z110 = Z - 110.0
      DTH = -2.291753 * Z110 + 0.02154336 * Z110*Z110- 4.1766671E-05 *
     $ (Z110 ** 3)
      DTH = EXP(-0.290655 * SQRT(ABS(Z110)))* DTH
      TH = TH +(DTH * C1 * C2 *TH) / 100.0
C
C     DENSITY IN METRIC UNITS AND PRESSURE CALCULATED
C
  300 DH = DH * 1000.0
      PH =((DH * 8.31432 * TH) / EM) * 1000.0
      RETURN
      END
      SUBROUTINE NORMAL(D1,D2)
C.....PRODUCES 2 RANDOM NUMBERS, D1, D2, PICKED FROM A NORMAL DIST.
C     WITH ZERO MEAN AND UNIT VARIANCE
      REAL L
      LOGICAL         OP    ,RST   ,HLD   ,RT    ,ATRM  ,ICEN  ,MDAT
      COMMON /RTCDAT/ OP    ,RST   ,HLD   ,RT    ,ATRM  ,ICEN  ,MDAT
C
C.... MODIFIED TO OUTPUT ZERO WHEN RESET FLAG IS TRUE.  THIS AVOIDS
C     OUTPUT BIASES INTRODUCED BY UNCHARACTERISTIC LARGE STEPS
C     INTRODUCED AT PROGRAM INITIALIZATION TIME OR WHEN IC'S CHANGE.
C
C.... MODIFIED   1/10/90   L SCHILLING   NASA/ADFRF.
C
   50 CONTINUE
C
      X = RAND(0)
      Y = 2*RAND(0) - 1
C
      XX = X**2
      YY = Y**2
      S = XX + YY
      IF (S.GT.1.0) GO TO 50
C
   51 CONTINUE
C
      L = SQRT(-2.0*ALOG(RAND(0)))/S
C
      D1 = (XX-YY)*L
      D2 = 2.0*X*Y*L
C
      IF(RST) THEN
         D1=0.0
     D2=0.0
      ENDIF
C
      RETURN
      END
      SUBROUTINE PDTUV (PSP, DSP, TSP, CLAT, CLON, IH, PS, DS, TS,
     $ DPX, DPY, DTX, DTY,DP2X,DP2Y,DPXY)
C
C.... DECLARE APPROPRIATE ARGUMENTS TO BE IN GOULD EXTENDED.  IF A
C     NON-EXTENDED PARAMETER IS DECLARED EXTENDED, THE INTERFACE WILL
C     WORK, ALBEIT WITH A SLIGHT EXECUTION OVERHEAD PENALTY.  LJS.
C
C.....INTERPOLATES STATIONARY PERTURBATIONS ON LATITUDE AND LONGITUDE
C        AT HEIGHT IH
      DIMENSION PSP(8,10,12),DSP(8,10,12),TSP(8,10,12)
      IF (IH.LT.52) GO TO 10
      IF (IH.GT.84) GO TO 20
C     HEIGHT INDEX K
      K = ((IH+4)/8) - 4
      GO TO 30
   10 K = (IH-20)/10
      GO TO 30
   20 K = 8
   30 XLON = CLON
      IF (CLON.LT.10.) XLON = 360. + CLON
C     LOWER LONGITUDE INDEX J
      J = INT((XLON + 20.)/30.)
C.....DLON - RELATIVE LONGITUDE DEVIATION FROM CORNER REFERENCE LOCATION
      DLON = (XLON - 30.*J + 20.)/30.
C     UPPER LONGITUDE INDEX JP
      JP = J+1
      IF (JP.GT.12) JP=1
C     LOWER LATITUDE INDEX I
      I = INT((CLAT + 110.)/20.)
C      UPPER LATITUDE INDEX IP
      IP = I+1
      IF (IP.GT.10) IP=10
C.....DLAT - RELATIVE LATITUDE DEVIATION FROM CORNER REFERENCE LOCATION
      DLAT = (CLAT-20.*I + 110.)/20.
C     PRESSURE LAT-LON INTERPOLATION
      PS=PSP(K,I,J)+(PSP(K,IP,J)-PSP(K,I,J))*DLAT+(PSP(K,I,JP)-PSP(K,I,J
     1))*DLON+(PSP(K,IP,JP)-PSP(K,I,JP)-PSP(K,IP,J)+PSP(K,I,J))*DLAT*
     2DLON
C     DENSITY LAT-LON INTERPOLATION
      DS=DSP(K,I,J)+(DSP(K,IP,J)-DSP(K,I,J))*DLAT+(DSP(K,I,JP)-DSP(K,I,J
     1))*DLON+(DSP(K,IP,JP)-DSP(K,I,JP)-DSP(K,IP,J)+DSP(K,I,J))*DLAT*
     2DLON
C     TEMPERATURE LAT-LON INTERPOLATION
      TS=TSP(K,I,J)+(TSP(K,IP,J)-TSP(K,I,J))*DLAT+(TSP(K,I,JP)-TSP(K,I,J
     1))*DLON+(TSP(K,IP,JP)-TSP(K,I,JP)-TSP(K,IP,J)+TSP(K,I,J))*DLAT*
     2DLON
C.....DPX - DP/DX FOR GEOSTROPHIC WINDS
      DPX = (PSP(K,I,J) - PSP(K,I,JP)) / 6.
      DPX = DPX + ((PSP(K,IP,J) - PSP(K,IP,JP))/6. - DPX)*DLAT
C.....DPY - DP/DY FOR GEOSTROPHIC WINDS
      DPY=(PSP(K,IP,J)-PSP(K,I,J))/4.
      DPY = DPY + ((PSP(K,IP,JP) - PSP(K,I,JP))/4. - DPY)*DLON
C.....DTX - DT/DX FOR THERMAL WINDS
      DTX = (TSP(K,I,J) - TSP(K,I,JP)) / 6.
      DTX = DTX + ((TSP(K,IP,J) - TSP(K,IP,JP))/6. - DTX)*DLAT
```

```
C.....DTY - DT/DY FOR THERMAL WINDS
      DTY = (TSP(K,IP,J) - TSP(K,I,J)) / 4.
      DTY = DTY + ((TSP(K,IP,JP) - TSP(K,I,JP))/4. - DTY)*DLON
      IF (IP.GT.9) GO TO 90
      DPXY = (PSP(K,IP,J) - PSP(K,IP,JP) - PSP(K,I,J) + PSP(K,I,JP))/24.
      JX = J - 1
      IF (JX.LT.1) JX = JX + 12
      IY = I -  1
      DP2X = (PSP(K,I,JX) - PSP(K,I,JP))/6.
      DP2X = DP2X + ((PSP(K,IP,JX) - PSP(K,IP,JP))/6. - DP2X)*DLAT
      DP2Y = (PSP(K,IP,J) - PSP(K,IY,J))/4.
      DP2Y = DP2Y + ((PSP(K,IP,JP) - PSP(K,IY,JP))/4.- DP2Y)*DLON
      RETURN
 90   DP2X = 0.
      DP2Y = 0.
      DPXY = 0.
      RETURN
      END
      SUBROUTINE PERTRB
C>>
      REAL*8 XCNT,PNDLX,RNDSX,RNTLX,RNTSX,RNULX,RNUSX,RNVLX,RNVSX
      COMMON /PLTOUT/ HWLS  ,HWLL  ,VDS   ,VTS   ,VUS   ,VDL   ,VTL   ,
     .                VUL   ,RDS   ,RTS   ,RVS   ,RDL   ,RTL   ,RVL   ,
     .                RNDL  ,RNDS  ,RNTL  ,RNTS  ,RNUL  ,RNUS  ,RNVL  ,
     .                RNVS  ,RNDLM ,RNDSM ,RNTLM ,RNTSM ,RNULM ,RNUSM ,
     .                RNVLM ,RNVSM
C>>
      COMMON/IOTEMP/IOTEM1,IOTEM2,IUG, IUN ,DD,XMJD,PHI1,PHI,NSAME,
     $PL1,DL1,TL1,SPL1,SDL1,STL1,UL1,VL1,SUL1,SVL1,MN,IDA,IYR,
     1PH,PLAT,
     * PLON,G,R,CH,CLAT,CLON,F10,F10B,AP,IHR,MIN,NMORE,DX,HL,VL,DZ,
     2B,EPS,IOPP,LOOK,IET,FLAT,PS1,DS1,TS1,US1,VS1,SPS1,SDS1,
     3STS1,SUS1,SVS1,UDS1,VDS1,UDL1,VDL1,UDS2,VDS2,UDL2,VDL2,DUMMY3(1)
      COMMON /COMPER/SP2,SD2,ST2,P2,D2,T2,U2,V2,SU2,SV2,CP,
     1PS2,DS2,TS2,US2,VS2,
     2PL2,DL2,TL2,UL2,VL2,
     3SPS2,SDS2,STS2,SUS2,SVS2,
     4SPL2,SDL2,STL2,SUL2,SVL2
      COMMON/WINCOM/ DUM(11),T,DUMMY2(9)
C>>
      DATA XCNT,RNDLX,RNDSX,RNTLX,RNTSX,RNULX,RNUSX,RNVLX,RNVSX /9*0.0/
C>>
      DLON = ABS(CLON-PLON)
      PI = 3.1415927
      IF(DLON.GT.PI) DLON = 2.*PI - DLON
      DX  =     R*SQRT((CLAT-PLAT)**2 + (COS(CLAT)*(DLON     ))**2)
C.....DX IS HORIZONTAL DISTANCE BETWEEN POSITIONS PLAT,PLON AND CLAT,CLO
      AH = 900.
      BH = 6.
C     HORIZONTAL WAVELENGTH, KM
      HLL= AH + BH*CH
C>>
      HWLL=HLL
C>>
      DPHI = (90. - ABS(CLAT)/0.017453293)**2
      DHGT = 0.22 + 0.00258*(SQRT(ABS(CH)**3))
      IF (DHGT.GT.5.) DHGT = 5.
      VDS = (11.0 - 2.102E-4*DPHI)*DHGT
      VTS = (3.0 + 5.146E-4*DPHI)*DHGT
      VUS = (6.2 - 3.615E-4*DPHI)*DHGT
      VDL = (20.7 - 1.346E-3*DPHI)*DHGT
      VTL = 7.3*DHGT
      VUL = (31.2 - 3.503E-3*DPHI)*DHGT
      HLS = 20. + .0125*CH*CH
      IF(HLS.GT.400.) HLS = 400.
C>>
      HWLS=HLS
C>>
      HLS = (DX/HLS)**2
      HLL = (DX/HLL)**2
      RDS=SQRT(HLS+(DZ/VDS)**2)
      IF(RDS.LE.100.)GO TO 10
      RDS=0.
      GO TO 20
   10  RDS=CORREL(RDS)
   20 RTS=SQRT(HLS+(DZ/VTS)**2)
      IF(RTS.LE.100.)GO TO 30
      RTS=0.
      GO TO 40
   30  RTS=CORREL(RTS)
   40 RVS=SQRT(HLS+(DZ/VUS)**2)
      IF(RVS.LE.100.)GO TO 50
      RVS=0.
      GO TO 60
   50  RVS=CORREL(RVS)
   60  RDL=SQRT(HLL+(DZ/VDL)**2)
      IF(RDL.LE.100.)GO TO 70
      RDL=0.
      GO TO 80
   70  RDL=CORREL(RDL)
   80 RTL=SQRT(HLL+(DZ/VTL)**2)
      IF(RTL.LE.100.)GO TO 90
      RTL=0.
      GO TO 100
   90  RTL=CORREL(RTL)
  100  RVL=SQRT(HLL+(DZ/VUL)**2)
      IF(RVL.LE.100.)GO TO 110
      RVL=0.
      GO TO 120
  110  RVL=CORREL(RVL)
  120  CONTINUE
      CALL CORLAT(AS,BS,CS,DS,ES,FS,GS,HS,AIS,AJS,AKS,SPS1,SPS2,SDS1,
     1 SDS2,STS1,STS2,SUS1,SUS2,SVS1,SVS2,UDS1,UDS2,VDS1,VDS2,RDS,RTS,
     2RVS)
      CALL CORLAT(AL,BL,CL,DL,EL,FL,GL,HL,AIL,AJL,AKL,SPL1,SPL2,SDL1,
     1 SDL2,STL1,STL2,SUL1,SUL2,SVL1,SVL2,UDL1,UDL2,VDL1,VDL2,
     2RDL,RTL,RVL)
      CALL NORMAL(ZD,ZT)
C>>
      XCNT=XCNT+1.0
      RNDS=ZD
      RNTS=ZT
      RNDSX=RNDSX+RNDS
      RNTSX=RNTSX+RNTS
      RNDSM=RNDSX/XCNT
      RNTSM=RNTSX/XCNT
C>>
      DS2=AS*DS1+BS*ZD
      TS2=CS*TS1+DS*DS2+ES*ZT
      FS2=DS2+TS2
      CALL NORMAL(ZD,ZT
C>>
      RNUS=ZD
      RNVS=ZT
      RNUSX=RNUSX+RNUS
      RNVSX=RNVSX+RNVS
      RNUSM=RNUSX/XCNT
      RNVSM=RNVSX/XCNT
```

```
C>>
      US2=FS*US1+GS*DS2+HS*ZD
      VS2=AIS*VS1+AJS*DS2+AKS*ZT
      CALL NORMAL(ZD,ZT)
C>>
      RNDL=ZD
      RNTL=ZT
      RNDLX=RNDLX+RNDL
      RNTLX=RNTLX+RNTL
      RNDLM=RNDLX/XCNT
      RNTLM=RNTLX/XCNT
C>>
      DL2=AL*DL1+BL*ZD
      TL2=CL*TL1+DL*DL2+EL*ZT
      PL2=DL2+TL2
      CALL NORMAL(ZD,ZT)
C>>
      RNUL=ZD
      RNVL=ZT
      PNULX=RNULX+RNUL
      RNVLX=RNVLX+RNVL
      RNULM=RNULX/XCNT
      RNVLM=RNVLX/XCNT
C>>
      UL2=FL*UL1+GL*DL2+HL*ZD
      VL2=AIL*VL1+AJL*DL2+AKL*ZT
      P2=PS2+PL2
      D2=DS2+DL2
      T2=TS2+TL2
      IF(P2.LT.-0.9)P2 = -0.9
      IF(D2.LT.-0.9)D2 = -0.9
      IF(T2.LT.-0.9)T2 = -0.9
      U2=US2+UL2
      V2=VS2+VL2
      UDL1=UDL2
      UDS1=UDS2
      VDL1=VDL2
      VDS1=VDS2
      RETURN
      END
      SUBROUTINE PHASE(D1,X1,D2,X2,D,X)
C
C.... DECLARE APPROPRIATE ARGUMENTS TO BE IN GOULD EXTENDED.  IF A
C     NON-EXTENDED PARAMETER IS DECLARED EXTENDED, THE INTERFACE WILL
C     WORK, ALBEIT WITH A SLIGHT EXECUTION OVERHEAD PENALTY.  LJS.
C
      PER = 870.
      IF (X2-X1) 20,10,20
 10   D = D1
      RETURN
 20   DA = D1
      DB = D2
      PER2 = PER/2.
      IF(ABS(DB-DA).LE.PER2)GO TO 30
      IF (DA.LT.PER2) DA = DA + PER
      IF (DB.LT.PER2) DB = DB + PER
   30 DA = DA + (DB - DA)*(X - X1)/(X2 - X1)
      IF (DA.GT.PER) DA = DA - PER
      IF(DA.LT.0.)DA=DA+PER
      D = DA
      RETURN
      END
      SUBROUTINE QBOGEN
C.....COMPUTES QBO VALUES PQ,DQ,TQ,UQ,VQ AT HEIGHT H, LATITUDE PHI
```

```
C          ON JULIAN DAY XMJD FROM ARRAYS OF AMPLITUDES PAQ,DAQ,TAQ,
C          UAQ,VAQ AND PHASES PDQ,DDQ,TDQ,UDQ,VDQ.
      COMMON/IOTEMP/IOTEM1,IOTEM2,IUG,IUN,DDD,XMJD,PHI1,PHI,
     .           NSAME,RP1, RD1, RT1, SP1, SD1, ST1, RU1, RV1, SU1, SV1,
     $ MN, IDA, IYR, H1, PHI1R,THET1R,G,RI,H,PHIR,THETR,F10,F10B,AP,
     $ IHR,MIN,NMORE,DX,HL,VL,DZ,DUMMY2(25)
      COMMON /PDTCOM/
     .               IU4,MONTH,IOPR,PG(18,19),TG(18,19),DG(18,19)
     . ,PSP(8,10,12)
     . ,DSP(8,10,12),TSP(8,10,12),PAQ(17,5),DAQ(17,5),TAQ(17,5),
     . PDQ(17,5),DDQ(17,5),TDQ(17,5),PR(20,10),DR(20,10),TR(20,10),
     .UAQ(17,5),VAQ(17,5),UDQ(17,5),VDQ(17,5),UR(25,10),VR(25,10)
     . ,PQ,DQ,TQ,UQ,VQ
     $ ,PA,DA,TA,UA,VA,IOPQ,DUMMY(2250)
      IF (XMJD.GT.0.AND.IOPQ.EQ.1) GO TO 10
C     SETS QBO VALUES TO ZERO FOR ANNUAL MEAN
      PQ =0.
      DQ=0.
      TQ=0.
      UQ=0.
      VQ=0.
      RETURN
C     LOWER HEIGHT INDEX
   10 IH = INT((H-5.)/5.)
      IF (IH.LT.1) IH=1
C     UPPER HEIGHT INDEX
      IP = IH + 1
      IF (IP.GT.17) IP = 17
      PHA = ABS(PHI)
C    LOWER LATITUDE INDEX

      JL = INT((  PHA    + 10.)/20.)
C     UPPER LATITUDE INDEX
      JP = JL + 1
      IF (JL.LE.0) JL=1
      IF (JP.GT.5) JP=5
C     JULIAN DAY FOR JAN 0, 1966
      XMJDO = 2439126
C     TIME RELATIVE TO JAN 0, 1966
      TMJD = XMJD-XMJDO
C     2*PI/PERIOD,PERIOD = 870 DAYS
      PER = 870.
      TP = 6.2831853/PER
C     LOWER HEIGHT
      HI = 5. + 5.*IH
C     LOWER LATITUDE
      PHIJ = 20.*JL - 10.
C     UPPER LATITUDE
      PHIP = 20.*JP-10.
C.....INTERPOLATES QBO P,D,T AMPLITUDE ON LATITUDE AT LOWER HEIGHT
      CALL INTERZ(PAQ(IH,JL),DAQ(IH,JL),TAQ(IH,JL),PHIJ,PAQ(IH,JP),
     1DAQ(IH,JP),TAQ(IH,JP),PHIP,PA1,DA1,TA1,PHA)
C     UPPER HEIGHT
      HP = 5.+5.*IP
C.....INTERPOLATES QBO P,D,T AMPLITUDE ON LATITUDE AT UPPER HEIGHT
      CALL INTERZ(PAQ(IP,JL),DAQ(IP,JL),TAQ(IP,JL),PHIJ,PAQ(IP,JP),
     2DAQ(IP,JP),TAQ(IP,JP),PHIP,PA2,DA2,TA2,PHA)
C.....INTERPOLATES QBO P,D,T AMPLITUDE ON HEIGHT AT LATITUDE PHI
      CALL INTERZ(PA1,DA1,TA1,HI,PA2,DA2,TA2,HP,PA,DA,TA,H)
C.....INTERPOLATES QBO P,D,T,U,V PHASE ON LATITUDE AND HEIGHT
      CALL PHASE(PDQ(IH,JL),PHIJ,PDQ(IH,JP),PHIP,PD1,PHA)
      CALL PHASE(DDQ(IH,JL),PHIJ,DDQ(IH,JP),PHIP,DD1,PHA)
      CALL PHASE(TDQ(IH,JL),PHIJ,TDQ(IH,JP),PHIP,TD1,PHA)
      CALL PHASE(PDQ(IP,JL),PHIJ,PDQ(IP,JP),PHIP,PD2,PHA)
      CALL PHASE(DDQ(IP,JL),PHIJ,DDQ(IP,JP),PHIP,DD2,PHA)
      CALL PHASE(TDQ(IP,JL),PHIJ,TDQ(IP,JP),PHIP,TD2,PHA)
      CALL PHASE(PD1,HI,PD2,HP,PD,H)
      CALL PHASE(DD1,HI,DD2,HP,DD,H)
      CALL PHASE(TD1,HI,TD2,HP,TD,H)
      CALL PHASE(UDQ(IH,JL),PHIJ,UDQ(IH,JP),PHIP,UD1,PHA)
      CALL PHASE(VDQ(IH,JL),PHIJ,VDQ(IH,JP),PHIP,VD1,PHA)
      CALL PHASE(UDQ(IP,JL),PHIJ,UDQ(IP,JP),PHIP,UD2,PHA)
      CALL PHASE(VDQ(IP,JL),PHIJ,VDQ(IP,JP),PHIP,VD2,PHA)
      CALL PHASE(UD1,HI,UD2,HP,UD,H)
      CALL PHASE(VD1,HI,VD2,HP,VD,H)
C.....INTERPOLATES QBO WIND AMPLITUDE ON LATITUDE AT LOWER HEIGHT
      CALL INTERW(UAQ(IH,JL),VAQ(IH,JL),PHIJ,UAQ(IH,JP),VAQ(IH,JP),
     5PHIP,UA1,VA1,PHA)
C.....INTERPOLATES QBO WIND AMPLITUDES ON LATITUDE AT UPPER HEIGHT
      CALL INTERW(UAQ(IP,JL),VAQ(IP,JL),PHIJ,UAQ(IP,JP),VAQ(IP,JP),
     6PHIP,UA2,VA2,PHA)
C.....INTERPOLATES QBO WIND AMPLITUDES ON HEIGHT AT LATITUDE PHI
      CALL INTERW(UA1,VA1,HI,UA2,VA2,HP,UA,VA,H)
C.....EVALUATES QBO VALUES FROM INTERPOLATED AMPLITUDES AND PHASES
      PQ=PA*COS(TP*(TMJD-PD))
      DQ=DA*COS(TP*(TMJD-DD))
      TQ=TA*COS(TP*(TMJD-TD))
      UQ=UA*COS(TP*(TMJD-UD))
      VQ=VA*COS(TP*(TMJD-VD))
      RETURN
      END
      FUNCTION RAND(X0)
C.....PRODUCES A RANDOM NUMBER FROM A UNIFORM DIST. FROM 0 TO +1
      INTEGER X0
      DOUBLE PRECISION X
      IF (X0.NE.0) X = X0/262144.
      X = X*509
      X = X - INT(X)
      RAND = X
      RETURN
      END
      SUBROUTINE RIG
      COMMON/IOTEMP/IOTEM1,IOTEM2,IUG,IUN,DD,XMJD,PHI1,PHI,
     .           NSAME,RP1, RD1, RT1, SP1, SD1, ST1, RU1, RV1, SU1, SV1,
     $ MN, IDA, IYR, H1, PHI1R,THET1R,G,RI,H,PHIR,THETR,F10,F10B,AP,
     $ IHR,MIN,NMORE,DX,HL,VL,DZ,B,EPS,IOPP,LOOK,IET,GLAT,
     1RP1S,RD1S,RT1S,RU1S,RV1S,SP1S,SD1S,ST1S,SU1S,SV1S,
     2UDS1,VDS1,UDL1,VDL1,UDS2,VDS2,UDL2,VDL2,DUMMY3(1)
C.....GRAVITY G AT H, LATITUDE PHIR (RADIANS)
C.....RADIUS RI FROM CENTER OF EARTH TO HEIGHT H
C.....B = POLAR EARTH RADIUS, EPS = ECCENTRICITY
      CPHI2 = COS(PHIR) ** 2
C      EARTH RADIUS
      RI = B / SQRT(1. - EPS * CPHI2)
C     C2PHI = COS(2*PHIR)
      C2PHI = 2. * CPHI2 - 1.
C     C4PHI = COS(4*PHIR)
      C4PHI = 8. * CPHI2 * (CPHI2 - 1.) + 1.
C.....G AT SURFACE
      G = 9.80616 * (1. - 0.0026373 * C2PHI + 0.0000059 * C2PHI * C2PHI)
C.....EFFECTIVE RADIUS
      RE = 2. * G / (3.085462E-3 + C2PHI * 2.27E-6 - C4PHI * 2.E-9)
C      G AT HEIGHT H
      G = G / (1. + (H / RE)) ** 2
C      RADIUS AT HEIGHT H
      RI = RI + H
      END
      SUBROUTINE RTERP(H,PHI,PR,DR,TR,P,D,T)
```

```
C
C.....COMPUTES RANDOM PERTURBATION STANDARD DEVIATIONS P,D,T AT
C        HEIGHT H (KM), LATITUDE PHI(DEGREES) FROM SIGMA ARRAYS
C        PR,DR,AND TR
C
C.... DECLARE APPROPRIATE ARGUMENTS TO BE IN COULD EXTENDED.  IF A
C     NON-EXTENDED PARAMETER IS DECLARED EXTENDED, THE INTERFACE WILL
C     WORK, ALBEIT WITH A SLIGHT EXECUTION OVERHEAD PENALTY.  LJS.
C
      DIMENSION PR(20,10),DR(20,10),TR(20,10)
C.....I = LOWER HEIGHT INDEX
      IF (H.LT.95.) I = INT((H-20.)/5.)
      IF (H.GE.95.) I = 14 + INT((H-80.)/20.)
      IF(I.LT.1)I=1
      IP = I+1
      IF (IP.GT.20) IP = 20
C      LOWER LATITUDE INDEX
      J = INT((PHI + 110.)/20.)
      JP = J+1
      IF (JP.GT.10) JP=10
      IF (I.GT.14) GO TO 10
C     LOWER HEIGHT FOR PR,TR,DR  ARRAYS
      Z1=5.*I+20.
      GO TO 20
   10 Z1=20.*(I-10)
   20 IF (IP.GT.14) GO TO 30
C     UPPER HEIGHT FOR PR,DR,TR ARRAYS
      Z2=5.*IP+20.
      GO TO 40
   30 Z2=20.*(IP-10)
   40 PHI1=-110.+20.*J
      PHI2=-110.+20.*JP
C.....INTERPOLATE ON LATITUDE AT LOWER HEIGHT
      CALL INTERZ(PR(I,J),DR(I,J),TR(I,J),PHI1,PR(I,JP),DR(I,JP),
     1            TR(I,JP),PHI2,P1,D1,T1,PHI)
C.....INTERPOLATE ON LATITUDE AT UPPER HEIGHT
      CALL INTERZ(PR(IP,J),DR(IP,J),TR(IP,J),PHI1,PR(IP,JP),DR(IP,JP),
     1            TR(IP,JP),PHI2,P2,D2,T2,PHI)
C.....INTERPOLATION ON HEIGHT USING LATITUDE INTERPOLATED VALUES
      CALL INTERZ(P1,D1,T1,Z1,P2,D2,T2,Z2,P,D,T,H)
      RETURN
      END
      SUBROUTINE SCIMOD(NPOP)
C.....COMPUTES VALUES P,D,T,U,V AND SHEAR DUH,DVH FROM INPUT AND
C        ARRAYS IN COMMON PDTCOM.  INPUT TO SCIMOD IS:
C        G = GRAVITY AT POSITION       RI = RADIUS AT HEIGHT H
C        PHIR = LATITUDE (RADIANS)     THETR = LONGITUDE (RADIANS)
C        F10 = F10.7 SOLAR FLUX        F10B = MEAN F10.7 FLUX
C        AP = SOLAR-GEOMAGNETIC A SUB P INDEX
C        MN/IDA/IYR = DATA (IYR = FULL YEAR-1900)
C        IHR%MIN = TIME                H1 = PREVIOUS HEIGHT
C        PHI1R = PREVIOUS LATITUDE     THET1R = PREVIOUS LONGITUDE
C        RP1,RD1,RT1 = PREVIOUS RANDOM PERTURBATIONS
C        SP1,SD1,ST1 = PREVIOUS RANDOM STANDARD DEVIATIONS (SIGMAS)
C        RU1,RV1 = PREVIOUS RANDOM WINDS
C        SU1,SV1 = PREVIOUS RANDOM WIND SIGMAS
      COMMON/IPRTP/ IPRT
      COMMON/IOTEMP/IOTEM1,IOTEM2,IUG, IUN ,DD,XMJD,PHI1,PHI,
     .NSAME,RP1L,RD1L,RT1L,SP1L,SD1L,ST1L,RU1L,RV1L,SU1L,SV1L,
     $ MN, IDA, IYR, H1, PHI1R,THET1R,G,RI,H,PHIR,THETR,F10,F10B,AP,
     .   IHR,MIN,NMORE,DX,HL,VL,DZ,B,EPS,IOFF,LOOK,IET,FLAT,
     1RP1S,RD1S,RT1S,RU1S,RV1S,SP1S,SD1S,ST1S,SU1S,SV1S,
     2UDS1,VDS1,UDL1,VDL1,UDS2,VDS2,UDL2,VDL2,DUMMY3(1)
      COMMON /PDTCOM/
     .               IU4,MONTH,IOPR,PG(18,19),TG(18,19),DG(18,19)
     . ,PSP(8,10,12)
     . ,DSP(8,10,12),TSP(8,10,12),PAQ(17,5),DAQ(17,5),TAQ(17,5),
     . PDQ(17,5),DDQ(17,5),TDQ(17,5),PR(20,10),DR(20,10),TR(20,10),
     .UAQ(17,5),VAQ(17,5),UDQ(17,5),VDQ(17,5),UR(25,10),VR(25,10), PQ
     . ,DQ,TQ,UQ,VQ,PQA,DQA,TQA,UA,VA,IOPQ,
     1PLP(25,10),DLP(25,10),TLP(25,10),
     2ULP(25,10),VLP(25,10),UDL(25,10),
     3VDL(25,10),UDS(25,10),VDS(25,10)
      COMMON /C4    /
     .             GLAT(16),GLON(16),NG,P4D(16,26),D4D(16,26),T4D(16,26),
     .  SP4(16,26),SD4(16,26),ST4(16,26),THET1,THET,DUMMY
      COMMON/COMPER/SPH,SDH,STH,PRH,DRH,TRH,URH,VRH,SUH,SVH,CP,
     1PRHS,DRHS,TRHS,URHS,VRHS,PRHL,DRHL,TRHL,URHL,VRHL,
     2SPHS,SDHS,STHS,SUHS,SVHS,SPHL,SDHL,STHL,SUHL,SVHL
       COMMON/WINCOM/DH,FCORY,DX5,DY5,DPX,DPY,DPXX,DPXY,DPYY,UGH,VGH,
     $     TH,DTX,DTY,DUH,DVH,PH ,UPRE,VPRE,DUPRE,DVPRE
      COMMON/CHIC/LA(4,4),NB(2),IWSYM,UCOEF(14,9),VCOEF(14,9)
      COMMON /GRAMOT/ PGH   ,DGH   ,TGH   ,UH    ,VH    ,PS    ,DS    ,
     .                TS    ,PGHP  ,DGHP  ,TGHP  ,PHP   ,DHP   ,THP   ,
     .                PSH   ,DSH   ,TSH   ,WGH
      REAL MOLWT
      DATA IBLK/1H /,IAST/1H*/
C      FACTOR FOR RADIANS TO DEGREES
      FAC = 57.2957795
      IWSYM = IBLK
      IF(NPOP.NE.0) GO TO 6
      UPRE=0.
      VPRE=0.
      DUPRE=0.
      DVPRE=0.
 6    PQ=0.
      DQ=0.
      TQ=0.
      PRH=0.
      DRH=0.
      TRH=0.
      URH=0.
      VRH=0.
      UQ=0.
      VQ=0.
      PQA=0.
      DQA=0.
      TQA=0.
      UA=0.
      VA=0.
      PSH=0.
      DSH=0.
      TSH=0.
      MONTH=MN
C     PRESENT LATITUDE, DEG
      PHI = PHIR*FAC
C     PRESENT LONGITUDE, DEG
      THET = THETR*FAC
C     PREVIOUS LATITUDE, DEG
      PHI1 = PHI1R*FAC
C     PREVIOUS LONGITUDE, DEG
      THET1 = THET1R*FAC
C.....FCORY = NORTH COMPONENT CORIOLIS FACTOR TIMES DISTANCE FOR
C        5 DEGREES OF LATITUDE
      DY5 = 5000.*RI/FAC
      DX5 = DY5*COS(PHIR)
      FCORY = DY5*SIN(PHIR)/(120.*FAC)
C.....IN JACCHIA OR MIXED GROVES-JACCHIA HEIGHT RANGE
```

```
    8 IF(H.GT.90.0) GO TO 10
C.....IN 4-D DATA HEIGHT RANGE
      IF (H.LE.25.0) GO TO 500
C       IN GROVES OR MIXED GROVES 4D HEIGHT RANGE
      GO TO 200
C.....IN MIXED JACCHIA-GROVES RANGE, NEED TO FAIR DATA
10    IF (H.LT.115.) GO TO 20
C.....FOLLOWING IS THE PURE JACCHIA HEIGHT RANGE SECTION
C.....JACCHIA VALUES AT CURRENT POSITION
      CALL JACCH(H,PHIR,THET,PH,DH,TH)
      PHIN = PHIR + 5. / FAC
      THETE = THET - 5.
C.....JACCHIA VALUES AT CURRENT POSITION+5 DEGREES LAT, FOR DP/DY AND
C         DT/DY
      CALL JACCH(H,PHIN,THET,PHN,DHN,THN)
C.....JACCHIA VALUES AT CURRENT POSITION-5 DEGREES LON, FOR DP/DX AND
C         DT/DX
      CALL JACCH(H,PHIR,THETE,PHE,DHE,THE)
C       DP/DY FOR GEOSTROPHIC WIND
      DPY=PHN-PH
C       DP/DX FOR GEOSTROPHIC WIND
      DPX=PHE-PH
C       DT/DX FOR THERMAL WIND SHEAR
      DTX = THE - TH
C       DT/DY FOR THERMAL WIND SHEAR
      DTY = THN - TH
C       CHANGE NOTATION FOR OUTPUT
      PGH=PH
      DGH=DH
      TGH=TH
      CALL WIND
      UH = UGH
      VH = VGH
      HB = H + 5.
      CP = 7.*PH/(2.*DH*TH)
      CALL JACCH(HB,PHIR,THET,PB,DB,TB)
      DTZ = (TB - TH)/5000.
C.....VERTICAL MEAN WIND
      WGH = -CP*(UH*DTX/DX5 + VH*DTY/DY5)/(G + CP*DTZ + UH*DUH+VH*DVH)
C       GO TO RANDOM PERTURBATIONS SECTION
      GO TO 800
C.....FOLLOWING IS THE MIXED JACCHIA-GROVES HEIGHT RANGE SECTION
C       LOWER HEIGHT INDEX
20    IHA = 5*(INT(H)/5)
C         UPPER HEIGHT INDEX
      IHB = IHA + 5
C     LOWER HEIGHT FOR INTERPOLATION
      HA = IHA*1.
C     UPPER HEIGHT FOR INTERPOLATION
      HB = IHB*1.
C.....JACCHIA VALUES AT LOWER HEIGHT, CURRENT LAT-LON
      CALL JACCH(HA,PHIR,THET,PJA,DJA,TJA)
      PHIN = PHIR + 5. / FAC
      THETE = THET - 5.
C.....JACCHIA VALUES AT LOWER HEIGHT, CURRENT LAT-LON+5 DEGREES
C         LAT, FOR DP/DY AND DT/DY
      CALL JACCH(HA,PHIN,THET,PJN,DJN,TJN)
C.....JACCHIA VALUES AT LOWER HEIGHT, CURRENT LAT-LON-5 DEGREES
C         LON, FOR DP/DX. AND DT/DX
      CALL JACCH(HA,PHIR,THETE,PJE,DJE,TJE)
C     JACCHIA DP/DY AT LOWER HEIGHT
      DPXJA=PJE-PJA
C       JACCHIA DP/DY AT LOWER HEIGHT
      DPYJA=PJN-PJA
```

```
C       JACCHIA DT/DX AT LOWER HEIGHT
      DTXJA = TJE - TJA
C       JACCHIA DT/DY AT LOWER HEIGHT
      DTYJA = TJN - TJA
C.....JACCHIA VALUES AT UPPER HEIGHT, CURRENT LAT-LON
      CALL JACCH(HB,PHIR,THET,PJB,DJB,TJB)
      PHIN = PHIR + 5. / FAC
      THETE=THETE-5
C.....JACCHIA VALUES AT UPPER HEIGHT, CURRENT LAT/LON+5 DEGREES
C         LAT, FOR DP/DY AND DT/DY
      CALL JACCH(HB,PHIN,THET,PJN,DJN,TJN)
C.....JACCHIA VALUES AT UPPER HEIGHT, CURRENT LAT-LON-5 DEGREES
C         LON, FOR DP/DX AND DT/DX
      CALL JACCH(HB,PHIR,THETE,PJE,DJE,TJE)
C     JACCHIA DP/DX FOR GEOSTROPHIC WINDS
      DPXJB = PJE - PJB
C      JACCHIA DP/DY FOR GEOSTOPHIC WINDS
      DPYJB = PJN - PJB
C     JACCHIA DT/DX FOR THERMAL WIND SHEAR
      DTXJB = TJE - TJB
C      JACCHIA DT/DY FOR THERMAL WIND SHEAR
      DTYJB = TJN - TJB
C.....GROVES AT LOWER HEIGHT, TO BE FAIRED WITH JACCHIA
      CALL GTERP(IHA,PHI,PGA,DGA,TGA,PG,DG,TG,DPYGA,DTYGA,DP2YGA)
C.....GROVES AT UPPER HEIGHT, TO BE FAIRED WITH JACCHIA
      CALL GTERP(IHB,PHI,PGB,DGB,TGB,PG,DG,TG,DPYGB,DTYGB,DP2YGB)
C.....FAIRED RESULTS AT LOWER HEIGHT
      IHSB = 90
      CALL PDTUV(PSP,DSP,TSP,PHI,THET,IHSB,PSH,DSH,TSH,DPXSB,
     $ DPYSB,DTXSB,DTYSB,DP2XSB,DP2YSB,DPXYSB)
      PGA = PGA*(1. + PSH)
      DGA = DGA*(1. + DSH)
      TGA = TGA*(1. + TSH)
      PGB = PGB*(1. + PSH)
      DGB = DGB*(1. + DSH)
      TGB = TGB*(1. + TSH)
      DTXGA = DTXSB * TGA
      DTXGB = DTXSB * TGB
      DTYGA = TGA*DTYSB + DTYGA*(1. + TSH + DTYSB)
      DTYGB = TGB*DTYSB + DTYGB*(1. + TSH + DTYSB)
      DPXGA = DPXSB * PGA
      DPXGB = DPXSB * PGB
      DPYGA = PGA*DPYSB + DPYGA*(1. + PSH + DPYSB)
      DPYGB = PGB*DPYSB + DPYGB*(1. + PSH + DPYSB)
      CALL FAIR(PGA,DGA,TGA,PJA,DJA,TJA,IHA,P1,D1,T1,DPXGA,DPYGA,
     $ DPXJA,DPYJA,DPXA,DPYA,DTXGA,DTYGA,DTXJA,DTYJA,DTXA,DTYA)
C.....FAIRED RESULTS AT UPPER HEIGHT
      CALL FAIR(PGB,DGB,TGB,PJB,DJB,TJB,IHB,P2,D2,T2,DPXGB,DPYGB,
     $DPXJB,DPYJB,DPXB,DPYB,DTXGB,DTYGB,DTXJB,DTYJB,DTXB,DTYB)
C.....HEIGHT INTERPOLATION ON FAIRED P,D,T
      CALL INTER2(P1,D1,T1,HA,P2,D2,T2,HB,PH,DH,TH,H)
C.....HEIGHT INTERPOLATION ON FAIRED DP/DX,DP/DY
      CALL INTERW(DPXA,DPYA,HA,DPXB,DPYB,HB,DPX,DPY,H)
C.....HEIGHT INTERPOLATION ON FAIRED DT/DX,DT/DY
      CALL INTERW(DTXA,DTYA,HA,DTXB,DTYB,HB,DTX,DTY,H)
C.....EASTWARD COMPONENT OF GEOSTROPHIC WIND
      CALL WIND
C      CHANGE OF VARIABLES FOR OUTPUT
      PGH=PH
      DGH=DH
      TGH=TH
      UH = UGH
      VH = VGH
      CP = 7.*PH/(2.*DH*TH)
      DTZ = (T2 - T1)/5000.
C.....VERTICAL MEAN WIND
      WGH = -CP*(UH*DTX/DX5 + VH*DTY/DY5)/(G + CP*DTZ + UH*DUH + VH*DVH)
C      GO TO RANDOM PERTURBATIONS SECTION
      GO TO 800
C.....THE FOLLOWING SECTION IS FOR GROVES OR MIXED GROVES 4D HEIGHTS
C      UPPER HEIGHT INDEX
 200  IHGB = 5*(INT(H)/5) + 5
      IF (IHGB.GT.90) IHGB=90
C     UPPER HEIGHT
      HGB = IHGB*1.
C.....GROVES AT UPPER HEIGHT
      CALL GTERP(IHGB,PHI,PGB,DGB,TGB,PG,DG,TG,DPYGB,DTYGB,DP2YGB)
C.....UPPER STATIONARY PERTURBATION HEIGHT = 40
      IF (H.LT.40.0) GO TO 210
C.....UPPER STATIONARY PERTURBATION HEIGHT = 90
      IF (H.GT.84.0) GO TO 220
C.....UPPER STATIONARY PERTURBATION HEIGHT = 52,60,68,76,OR 84
      IHSB = 8*((INT(H) + 4)/8) + 4
C.....UPPER STATIONARY PERTURBATION HEIGHT = 52
      IF (IHSB.LT.52.0) IHSB = 52
      GO TO 230
  210 IHSB = 10*(INT(H)/10) + 10
      GO TO 230
 220  IHSB = 90
C     UPPER STATIONARY PERTURBATION HEIGHT
 230  HSB = IHSB*1.
C.....STATIONARY PERTURBATIONS AT UPPER HEIGHT
      CALL PDTUV(PSP,DSP,TSP,PHI,THET,IHSB,PSB,DSB,TSB,DPXSB,DPYSB,
     $ DTXSB,DTYSB,DP2XSB,DP2YSB,DPXYSB)
C      MIXED GROVES 4D SECTION
      IF (H.LT.30.0) GO TO 300
C     LOWER HEIGHT INDEX
      IHGA = IHGB - 5
C     LOWER HEIGHT INDEX
      HGA = IHGA*1.
C.....GROVES AT LOWER HEIGHT
      CALL GTERP(IHGA,PHI,PGA,DGA,TGA,PG,DG,TG,DPYGA,DTYGA,DP2YGA)
C.....LOWER STATIONARY PERTURBATION HEIGHT = 30
      IF (H.LT.40.0) GO TO 240
C.....LOWER STATIONARY PERTURBATION HEIGHT = 52,60,68,76, OR 84
      IHSA = 8*((INT(H) + 4)/8) - 4
C.....LOWER STATIONARY PERTURBATIONS HEIGHT = 40
      IF (IHSA.LT.52.0) IHSA = 40
      GO TO 250
  240 IHSA = 30
C     LOWER STATIONARY PERTURBATION HEIGHT
 250  HSA = IHSA*1.
C.....STATIONARY PERTURBATIONS AT LOWER HEIGHT
      CALL PDTUV(PSP,DSP,TSP,PHI,THET,IHSA,PSA,DSA,TSA,DPXSA,DPYSA,
     $ DTXSA,DTYSA,DP2XSA,DP2YSA,DPXYSA)
C.....GROVES VALUES HEIGHT INTERPOLATIONS
      CALL INTER2(PGA,DGA,TGA,HGA,PGB,DGB,TGB,HGB,PGH,DGH,TGH,H)
C.....STATIONARY PERTURBATION HEIGHT INTERPOLATION
      CALL INTERZ(PSA,DSA,TSA,HSA,PSB,DSB,TSB,HSB,PSH,DSH,TSH,H)
C     QUASI-BIENNIAL VALUES
      CALL QBOGEN
C.....HEIGHT INTERPOLATION OF GROVES DP/DY, DT/DY, AND D2P/DY2
      CALL INTERZ(DPYGA,DTYGA,DP2YGA,HGA,DPYGB,DTYGB,DP2YGB,HGB,DPYG,
     $   DTYG,DP2YG,H)
C.....HEIGHT INTERPOLATION OF STATIONARY PERTURBATION DP/DX AND DP/DY
      CALL INTERW(DPXSA,DPYSA,HSA,DPXSB,DPYSB,HSB,DPXS,DPYS,H)
C.....HEIGHT INTERPOLATION OF STATIONARY PERTURBATION DT/DX AND DT/DY
      CALL INTERW(DTXSA,DTYSA,HSA,DTXSB,DTYSB,HSB,DTXS,DTYS,H)
```

```
C.....HEIGHT INTERPOLATION OF STATIONARY PERTURBATION D2P/DX2,D2P/DY2,
C                AND D2P/DXDY
      CALL INTERZ(DP2XSA,DP2YSA,DPXYSA,HSA,DP2XSB,DP2YSB,DPXYSB,HSB,
     $       DP2XS,DP2YS,DPXYS,H)
C.....UNPERTURBED (MONTHLY MEAN) VALUES FOR OUTPUT
      TGH = TGH * (1. + TSH)
      PGH = PGH * (1. + PSH)
      DGH = DGH * (1. + DSH)
C     TOTAL DT/DX
      DTX =         DTXS * TGH
C     TOTAL DT/DY
      DTY = TGH*DTYS + DTYG*(1. + TSH + DTYS)
C     TOTAL DP/DX
      DPX =         DPXS * PGH
C     TOTAL DP/DY
      DPY = PGH*DPYS + DPYG*(1. + PSH + DPYS)
C     D2P/DX2
      DPXX = PGH*(2.*DPXS - DP2XS)
      DPYY = PGH*(2.*DPYS - DP2YS) +  (2.*DPYG - DP2YG)*(1. +PSH+DPYS)
     $     - (DPYG - DP2YG)*DP2YS
C     D2P/DXDY
      DPXY = (PGH + DPYG)*DPXYS + DPYG*DPXS
C.....UNPERTURBED VALUES PLUS QBO PERTURBATIONS
      PH = (1. + PQ) * PGH
      DH = DGH * (1. + DQ)
      TH = (1. + TQ) * TGH
      CALL WIND
C     GEOSTROPHIC WIND PLUS QBO WIND PERTURBATIONS
      UH=UGH+UQ
      VH=VGH+VQ
      CP = 7.*PGH/(2.*DGH*TGH)
      DTZ = (TGB*(1.+TSB) - TGA*(1.+TSA))/5000.
C.....VERTICAL MEAN WIND
      WGH=-CP*(UGH*DTX/DX5+VGH*DTY/DY5)/(G+CP*DTZ+VGH*DUH+VGH*DVH)
C      GO TO RANDOM PERTURBATIONS SECTION
      GO TO 800
C.....THE FOLLOWING IS THE MIXED GROVES 4D SECTION
C.....GENERATE GRID OF 4D PROFILES IF PREVIOUS HEIGHT GE 30
 300   IF (H1.GE.30..OR.LOOK.EQ.1) CALL GEN4D
C300   IF (H1.GE.30..OR.LOOK.EQ.1) CALL USGRID
C
 310  CONTINUE
C.....LAT-LON INTERPOLATION OF 4D DATA AT 25 KM
      CALL INTER4(          PHI,THET,25,    P4D,D4D,T4D,P4A,D4A,T4A,
     $ DPX4,DPY4,DTX4,DTY4,DPXXA,DPYYA,DPXYA)
C      GROVES PLUS STATIONARY PERTURBATIONS
      PB = PGB*(1. + PSB)
C      P,D,T
      DB = DGB*(1. + DSB)
      TB = TGB*(1. + TSB)
      DPXB = PGB*DPXSB
      DPYB = PGB*DPYSB + DPYGB*(1. + PSB + DPYSB)
      DPXXB = PGB*(2.*DPXSB - DP2XSB)
      DPYYB = PGB*(2.*DPYSB - DP2YSB) + (2.*DPYGB - DP2YGB)*
     $  (1. + PSB + DPYSB) - (DPYGB - DP2YGB)*DP2YSB
      DPXYB = (PGB + DPYGB)*DPXYSB + DPYGB*DPXSB
      DTXB = TGB*DTXSB
      DTYB = TGB*DTYSB + DTYGB*(1. + TSB + DTYSB)
C.....HEIGHT INTERPOLATION BETWEEN 4D AT 25 AND GROVES AT UPPER HEIGHT
C          DP/DX AND DP/DY
      CALL INTERW(DPX4,DPY4,25.,DPXB,DPYB,HSB,DPX,DPY,H)
C.....HEIGHT INTERPOLATION BETWEEN 4D AT 25 AND GROVES AT UPPER HEIGHT
C          P,D,T
      CALL INTER2(P4A,D4A,T4A,25.,PB,DB,TB,HGB,PGH,DGH,TGH,H)
```

```
C.....HEIGHT INTERPOLATION BETWEEN 4D AT 25 AND GROVES AT UPPER HEIGHT
C        DT/DX AND DT/DY
      CALL INTERW(DTX4,DTY4,25.,DTXB,DTYB,HSB,DTX,DTY,H)
C.....HEIGHT INTERPOLATION BETWEEN 4D AT 25 KM AND GROVES AT UPPER
C     HEIGHT D2P/DX2, D2P/DY2, AND D2P/DXDY
      CALL INTERZ(DPXXA,DPYYA,DPXYA,25.,DPXXB,DPYYB,DPXYB,HGB,DPXX,
     $      DPYY,DPXY,H)
      IF (IOPQ.EQ.2) GO TO 350
C     QUASI BIENNIAL PERTURBATIONS
      CALL QBOGEN
C     ADD QBO PERTURBATIONS TO P,D,T
  350 PH=PGH*(1.+PQ)
      DH=DGH*(1.+DQ)
      TH=TGH*(1.+TQ)
      CALL WIND
C     ADD QBO WIND PERTURBATIONS
      UH=UGH+UQ
      VH=VGH+VQ
      CP = 7.*PGH/(2.*DGH*TGH)
      DTZ = (TB - T4A)/(1000.*(HGB - 25.))
C.....VERTICAL MEAN WIND
      WGH=-CP*(UGH*DTX/DX5+VGH*DTY/DY5)/(G+CP*DTZ+UGH*DUH+VGH*DVH)
C     GO TO RANDOM PERTURBATIONS SECTION
C:: 2000 FORMAT(1H ,'LATITUDE',/16F8.3)
C:: 2001 FORMAT(1H ,'LONGITUDE',/16F8.3,/' PRESSURE')
C:: 2002 FORMAT(1X,I2,16F8.0)
      GO TO 800
  500 IF (H.GE.0.0) GO TO 510
      IF (H.LT.-0.015) GO TO 505
C     IF -15 METER LE H LT 0 , H IS SET TO 0
      H = 0.
      GO TO 510
C     NO MORE COMPUTATIONS TO BE MADE IF HEIGHT LT -5 M
  505 NMORE = 0
       RETURN
C.....GENERATE GRID OF 4D PROFILES IF PREVIOUS HEIGHT GE 30
  510 IF (H1.GE.30..OR.LOOK.EQ.1) CALL GEN4D
C510   IF (H1.GE.30..OR.LOOK.EQ.1) CALL USGRID
C      LOWER HEIGHT INDEX
       IHA=INT(H)
C      LOWER HEIGHT INDEX
       HA = IHA*1.
      IWSX = IWSYM
C      UPPER HEIGHT INDEX
       IHB = IHA + 1
       IF(IHB.LE.25) GO TO 513
       IHA=24
       HA=24.
       IHB=25
C       UPPER HEIGHT
  513 HB = IHB*1.
C.....LAT-LON INTERPOLATION OF 4D VALUES AT UPPER HEIGHT
  515 CALL INTER4(          PHI,THET,IHB,   P4D,D4D,T4D,PB,DB,TB,
     $ DPX4B,DPY4B,DTX4B,DTY4B,DPXXB,DPYYB,DPXYB)
      IF(IHA.EQ.0.AND.PB*DB*TB.LE.0.)GO TO 520
      GO TO 540
  520 IHB=IHB+1
C.....LOOP TO FIND LOWEST VALID HEIGHT
      HB=HB+1.
      GO TO 515
  540 IF(IHA.GT.0) CALL INTER4(          PHI,THET,IHA.   P4D,D4D,T4D,
     1PA,DA,TA,DPX4A,DPY4A,DTX4A,DTY4A,DPXXA,DPYYA,DPXYA)
      IF (IWSYM .EQ.IAST) IWSX = IWSYM
      IF(IHA.EQ.0.OR.(PA*DA*TA.LE.0.AND.IHA.LT.10.AND.PB*DB*TB.GT.0.))
     1GO TO 550
      GO TO 600
C.....LAT-LON INTERPOLATION OF 4D VALUES AT LOWER HEIGHT
  550 CALL INTER4(          PHI,THET,0,   P4D,D4D,T4D,
     .PA,DA,TA,DPX4A,DPY4A,DTX4A,DTY4A,DPXXA,DPYYA,DPXYA)
      IF (IWSYM .EQ. IAST) IWSX = IWSYM
      IF(TA-TB)560,570,560
  560 IF(TA*TB.LE.0.0)GO TO 570
      TZ=(TA-TB)/ALOG(TA/TB)
      GO TO 575
  570 TZ=TA
C ...COMPUTES HEIGHT OF SURFACE
  575 HA=HB
      IF(PB*PA.LE.0.0)GO TO 576
      HA=HB+0.28705*TZ*ALOG(PB/PA)/G
  576 IF(H.GT.HA-0.04)GO TO 600
      PH=0.
      DH=0.
      TH=0.
      PGH=0.
      DGH=0.
      TGH=0.
      GO TO 800
C.....HEIGHT INTERPOLATION OF P,D,T
  600 CALL INTER2(PA,DA,TA,HA,PB,DB,TB,HB,PGH,DGH,TGH,H)
C.....HEIGHT INTERPOLATION OF DP/DX AND DP/DY
      CALL INTERW(DPX4A,DPY4A,HA,DPX4B,DPY4B,HB,DPX,DPY,H)
C.....HEIGHT INTERPOLATION OF DT/DX AND DT/DY
      CALL INTERW(DTX4A,DTY4A,HA,DTX4B,DTY4B,HB,DTX,DTY,H)
C.....HEIGHT INTERPOLATION OF D2P/DX2, D2P/DY2, AND D2P/DXDY
      CALL INTERZ(DPXXA,DPYYA,DPXYA,HA,DPXXB,DPYYB,DPXYB,HB,DPXX,DPYY,
     $DPXY,H)
C       CHANGE OF NOTATION FOR OUTPUT
      PH = PGH
      DH = DGH
      TH = TGH
      IF(PH*DH*TH.LE.0.) GO TO 800
      CALL WIND
C       CHANGE OF NOTATION FOR OUTPUT
      UH = UGH
      VH = VGH
      CP = 7.*PGH/(2.*DGH*TGH)
      DTZ = (TB - TA)/(1000.*(HB - HA))
C.... VERTICAL MEAN WIND
      WGH = -CP*(UGH*DTX/DX5 + VGH*DTY/DY5)/(G+CP*DTZ+UH*DUH+VH*DVH)
C     QBO=0 IF H LT 10
      IF (H.LT.10.) GO TO 800
      IF (IOPQ.EQ.2) GO TO 650
C     COMPUTES QUASI BIENNIAL PERTURBATIONS
      CALL QBOGEN
C     ADDS QBO PERTURBATIONS TO P,D,T
  650 PH=PGH*(1.+PQ)
      DH=DGH*(1.+DQ)
      TH=TGH*(1.+TQ)
C     ADDS QBO WIND PERTURBATIONS TO U,V
      UH=UGH+UQ
      VH=VGH+VQ
C.....THE FOLLOWING IS THE RANDOM PERTURBATIONS SECTION
C.....NO RANDOM PERTURBATIONS IF IOPP GT 1
  800 CONTINUE
      IF(H.GT.30) GO TO 512
      IF (IPRT .NE. 0 .OR. IWSYM .NE. IAST) GO TO 512
C::   WRITE(6,2000) (GLAT(I),I=1,16)
C::   WRITE(6,2001) (GLON(1),I=1,16)
```

```
C::   DO 504 I=1,26
C::   IH=I-1
C::   WRITE(6,2002) IH,(P4D(J,I),J=1,16)
C::  504 CONTINUE
      IPRT=IPRT+1
  512 CONTINUE
      IF (IOPR.GT.1) GO TO 830
      IF(NPOP.EQ.0)GO TO 840
C.....INTERPOLATES RANDOM WIND MAGNITUDES TO HEIGHT H, LATITUDE PHI
      CALL INTRUV(UR,VR,H,PHI,SUH,SVH)
      CALL INTRUV(PLP,DLP,H,PHI,PLPH,DLPH)
      CALL INTRUV(TLP,DLP,H,PHI,TLPH,DLPH)
      CALL INTRUV(ULP,VLP,H,PHI,ULPH,VLPH)
      CALL INTRUV(UDL,VDL,H,PHI,UDL2,VDL2)
      CALL INTRUV(UDS,VDS,H,PHI,UDS2,VDS2)
      SUHL=SQRT(ULPH*ABS(SUH))
      SUHS=SQRT((1.-ULPH)*ABS(SUH))
      SVHL=SQRT(VLPH*ABS(SVH))
      SVHS=SQRT((1.-VLPH)*ABS(SVH))
      SUH = SQRT(ABS(SUH))
      SVH = SQRT(ABS(SVH))
      IF(H.GE.25.)GO TO 805
C.....IF H LE 20 USE 4D DATA RANDOM P,D,T SIGMAS
      IF (H.LE.20.) GO TO 810
C.....INTERPOLATE PR,DR,TR ARRAYS TO GET P,D,T SIGMAS AT HEIGHT H,
C        LATITUDE PHI
      CALL RTERP(25.,PHI,PR,DR,TR,SPHG,SDHG,STHG)
      GO TO 810
  805 CONTINUE
      CALL RTERP(H,PHI,PR,DR,TR,SPH,SDH,STH)
      GO TO 820
C.....LAT-LON INTERPOLATION ON P,D,T SIGMAS AT LOWER HEIGHT
  810 CALL INTER4(          PHI,THET,IHA,   SP4,SD4,ST4,PA,DA,TA,
     $ DPX,DPY,DTX,DTY,DPXX,DPYY,DPXY)
C.....LAT-LON INTERPOLATION ON P,D,T SIGMAS AT UPPER HEIGHT
      CALL INTER4(          PHI,THET,IHB,   SP4,SD4,ST4,PB,DB,TB,
     $ DPX,DPY,DTX,DTY,DPXX,DPYY,DPXY)
C.....HEIGHT INTERPOLATION OF SIGMAS
      CALL INTERZ(PA,DA,TA,   HA,PB,DB,TB,   HB,SPH,SDH,STH,H)
      IF(SPH.LE.0.0.OR.SDH.LE.0.0.OR.STH.LE.0.0)GO TO 825
      IF(PH.LE.0.0.OR.DH.LE.0.0.OR.TH.LE.0.0)GO TO 825
      IF(H.LE.20.)GO TO 820
      FH = 1. - 0.2*(25. - H)
      SPH = FH*SPHG + (1. - FH)*SPH
      SDH = FH*SDHG + (1. - FH)*SDH
      STH = FH*STHG + (1. - FH)*STH
C.....HEIGHT DISPLACEMENT BETWEEN PREVIOUS AND CURRENT POSITION
  820 DZ = H1 - H
      SPHL=SQRT(PLPH*ABS(SPH))
      SPHS=SQRT((1.-PLPH)*ABS(SPH))
      SDHL=SQRT(DLPH*ABS(SDH))
      SDHS=SQRT((1.-DLPH)*ABS(SDH))
      STHL=SQRT(TLPH*ABS(STH))
      STHS=SQRT((1.-TLPH)*ABS(STH))
      SPH = SQRT(ABS(SPH))
      SDH = SQRT(ABS(SDH))
      STH = SQRT(ABS(STH))
C.....COMPUTES HORIZONTAL DISPLACEMENT DX BETWEEN PREVIOUS AND CURRENT
C        POSITION, HORIZONTAL SCALE HL, AND VERTICAL SCALE VL
C.....COMPUTES PERTURBATION VALUES PPH,DPH,TPH,UPH AND VPH
      CALL PERTRB
C     ADDS RANDOM PERTURBATIONS TO PH,DH,TH
      PH = PH*(1. + PPH)
      DH = DH*(1. + DPH)
      TH = TH*(1. + TRH)
C     ADDS RANDOM WINDS TO UH,VH
      UH=UH+URH
      VH=VH+VRH
C.....SETS PREVIOUS RANDOM PERTURBATION IN P,D,T TO CURRENT
C          PERTURBATIONS, FOR NEXT CYCLE
  825 RP1S= PRHS
      RD1S= DRHS
      RT1S= TRHS
      RP1L=PRHL
      RD1L=DRHL
      RT1L=TRHL
C.....SETS PREVIOUS MAGNITUDES FO CURRENT VALUES, FOR NEXT CYCLE
      SP1S=SPHS
      SD1S= SDHS
      ST1S=STHS
      SP1L=SPHL
      SD1L=SDHL
      ST1L=STHL
C.....SETS PREVIOUS WIND PERTURBATION VALUES TO CURRENT VALUES,
C          FOR NEXT CYCLE
      RU1S=URHS
      RV1S=VRHS
      RU1L=URHL
      RV1L=VRHL
C.....SETS PREVIOUS WIND PERTURBATION MAGNITUDES TO CURRENT VALUES,
C          FOR NEXT CYCLE
      SU1S=SUHS
      SV1S=SVHS
      SU1L=SUHL
      SV1L=SVHL
C.....SETS PREVIOUS HEIGHT TO CURRENT HEIGHT, FOR NEXT CYCLE
  830 H1 = H
C.....SETS PREVIOUS LATITUDE TO CURRENT LATITUDE, FOR NEXT CYCLE
      PHI1R=PHIR
C.....SETS PREVIOUS LONGITUDE TO CURRENT LONGITUDE, FOR NEXT CYCLE
      THET1R=THETR
C     SETS NMORE TO COMPUTE MORE DATA ON NEXT CYCLE
  840 NMORE = 1
C.....NO MORE DATA IF P, D, OR T LEQ 0
      IF(PH*DH*TH.LE.0.) RETURN
      CALL STDATM(H,TS,PS,DS)
      IF ((PS*DS*TS).GT.0.) GO TO 870
      PGHP=0.
      DGHP=0.
      TGHP=0.
      PHP=0.
      DHP=0.
      THP=0.
      GO TO 880
  870 PGHP=100.*(PGH-PS)/PS
      DGHP=100.*(DGH-DS)/DS
      TGHP=100.*(TGH-TS)/TS
      PHP=100.*(PH-PS)/PS
      DHP=100.*(DH-DS)/DS
      THP=100.*(TH-TS)/TS
C     CONVERTS OBS P,D,T TO PERCENT
  880 PQ=100.*PQ
      DQ=100.*DQ
      TQ=100.*TQ
C     CONVERTS RANDOM P,D,T TO PERCENT
      PRH=100.*PRH
      DRH=100.*DRH
      TRH=100.*TRH
      PRHS=100.*PRHS
      DRHS=100.*DRHS
      TRHS=100.*TRHS
      PRHL=100.*PRHL
      DRHL=100.*DRHL
      TRHL=100.*TRHL
      SPHS = 100.*SPHS
      SDHS = 100.*SDHS
      STHS = 100.*STHS
      SPHL = 100.*SPHL
      SDHL = 100.*SDHL
      STHL = 100.*STHL
C     CONVERTS WIND SHEAR TO M/S/KM
      DUH = DUH * 1000.
      DVH = DVH * 1000.
C      CONVERTS VERTICAL WIND TO CM/S
      WGH = WGH*100.
      PQA=PQA*100.
      DQA=DQA*100.
      TQA=TQA*100.
      SPH=SPH*100.
      SDH=SDH*100.
      STH=STH*100.
      PSH=PSH*100.
      DSH=DSH*100.
      TSH=TSH*100.
      IF(NPOP.EQ.0) THEN
         UPRE=UGH
         VPRE=VGH
         DUPRE=DUH/1000.
         DVPRE=DVH/1000.
      ENDIF
C
      RETURN
       ND
      SUBROUTINE SETUP
C
C.... SETUP HAS BEEN MODIFIED TO READ ALL INPUT FROM FILES WHICH
C     ARE OPENED THEN CLOSED TO MINIMIZE I/O BUFFER SPACE REQUIRED
C     BY THE PROGRAM.  AS SUCH, ALL PREDEFINED LUN'S ARE IGNORED.
C
C.... THE ORIGINAL SETUP ROUTINE ZEROED THE RANDOM PERTURBATIONS IF
C     IOPR=2 AND ZEROED THE QUASI-BIENNIAL OSCILLATIONS IF IOPQ=2.
C     THIS VERSION OF THE PROGRAM READS IN THE DATA REGARDLESS, THEN
C     DISABLES THE RP'S AND THE QBO'S WHRE THEIR EFFECT IS SUMMED IN.
C     THIS ALLOWS REAL-TIME CONTROL OF THESE FEATURES.
C
C.... WRITTEN   26 JAN 89   L SCHILLING   NASA/ADFRF.
C
      CHARACTER*12 FILNAM
      DIMENSION IB(5),ID(5),IT(5),IDAY(12)
      DIMENSION IDUM[illegible]
      DIMENSION NDATA[illegible],IX(10)
C
      COMMON [illegible] [illegible],IOTEM,[illegible] ,[illegible] ,[illegible]  XMU  ,PHI1
     .                 [illegible]  ,NSAMP ,PFII  ,PI[illegible] ,P[illegible] ,SFII ,SI[illegible]  ,
     .                 [illegible]  ,PU11  ,PV11  ,[illegible] ,[illegible]11  ,NS  ,IDI   ,
     .                 [illegible]  ,HI    ,PHTIP ,THETIP,[illegible]MS(21),PF1C,PI1C  ,
     .                 [illegible]  ,PU1[illegible] ,PV1[illegible] ,[illegible] ,[illegible] ,[illegible] ,[illegible]1S  ,
     .                 [illegible]  ,[illegible] ,[illegible] ,[illegible] ,[illegible] ,[illegible] ,[illegible]  ,
     .                 [illegible]  ,[illegible] ,[illegible]
       MMON [illegible]
     .                 [illegible]4   MONTH ,[illegible]FF  ,PG(18,19),TG(18,19),
     .                 [illegible]18,19),PSP(8,10,12),DSP(8,1 ,12),TSP(8,10,1[illegible]),
     .                  PAQ(17,5),DAQ(17,5),TAQ(17,5),PDQ(17,5),
     .                  DDQ(17,5),TDQ(17,5),PR(20,10),DR(20,10),
     .                  TR(20,10),UAQ(17,5),VAQ(17,5),UDQ(17,5),
     .                  VDQ(17,5),UR(25,10),VR(25,10),PQ     ,DQ     ,
     .                  TQ     ,UQ     ,VQ     ,PQA    ,DQA    ,TQA    ,UA     ,
     .                  VA     ,IOPQ   ,PLP(25,10),DLP(25,10),TLP(25,10),
     .                  ULP(25,10),VLP(25,10),UDL(25,10),VDL(25,10),
     .                  UDS(25,10),VDS(25,10)
      COMMON /CHIC   / DUM(18),IWSYM,UCOEF(14,9),VCOEF(14,9)
      DATA IDAY /  0, 31, 59, 90,120,151,181,212,243,273,304,334/
C
      XMJD = 0.
      IF (MN.GT.12) GO TO 2
      IDA = IDAY(MN) + IDD
      DD = IDA
      IF (MOD(IYR,4).EQ.0.AND.MN.GT.2) IDA = IDA + 1
      XMJD = 2439856.0 + 365.0*(IYR-68.) + IDA + INT((IYR-65.0)/4.0)
C
C.... SECOND DATA CARD READS. FREE FIELD, LOGICAL UNIT NUMBERS FOR
C     THE GRAM PROGRAM.  THESE LUNS ARE IGNORED IN THE OPEN FILE,
C     READ FILE, CLOSE FILE APPROACH.  PROGRAM OPTIONS ARE ALSO READ
C     IN.  THEY ARE DEFINE AS FOLLOWS:
C
C        IOPR=1:  RANDOM OUTPUT, =2:  NO RANDOM OUTPUT
C        IOPQ=1:  QBO OUTPUT    , =2:  NO QBO OUTPUT
C        NR1 = STARTING RANDOM NUMBER
C
    2 READ(55,*) TUN   ,IUG   ,IUR   ,IUVC  ,IUQ   ,IUS   ,
     .           IU4   ,IOPR  ,IOPQ  ,NR1   ,IOTEM1,IOTEM2
C
      IF (IOPR.LT.1.OR.IOPR.GT.2) GO TO 666
      IF (IOPQ.LT.1.OR.IOPQ.GT.2) GO TO 666
C
      MONTH=MN
      RPSCALE = 1.0
C:    R = RAND(NR1)           ! INIT CALLS MADE IN 'GRAMRT' IN
C:    R = RAND(0)             ! REAL-TIME VERSION.  VALUE OF 1.0
C:    R = RAND(0)             ! IS ASSUMED FOR NR1 SINCE NOT IN COMMON
C
C.... THIRD DATA CARD READS FREE FIELD, THE FOLLOWING DATA:
C        RP1 = INITIAL RANDOM PRESSURE PERTURBAIIONS, PERCENT
C        RD1 = INITIAL RANDOM DENSITY PERTURBATION, PERCENT
C        RT1 = INITIAL RANDOM TEMPERATURE PERTURBATION, PERCENT
C        SD1 = INITIAL STANDARD DEVIATION FOR RANDOM DENSITY
C                  PERTURBATION, PERCENT
C        RU1 = INITIAL EASTWARD WIND PERTURBATION, M/S
C        RV1 = INITIAL NORTHWARD WIND PERTURBATION, M/S
C        SU1 = INITIAL STANDARD DEVIATION FOR RANDOM EASTWARD WIND, M/S
C        SV1 = INITIAL STANDARD DEVIATION FOR RANDOM NORTHWARD WIND, M/S
C
      READ(55,*) RP1L  ,RP1S  ,RD1L  ,RD1S  ,RT1L  ,RT1S  ,
     .           RU1L  ,RU1S  ,RV1L  ,RV1S  ,RPSCALE
      IF (RPSCALE.LT.0.0 .OR. RPSCALE.GT.2.0) RPSCALE=1.0
      RP1=RP1L+RP1S
      RD1=RD1S+RD1L
      RT1=RT1S+RT1L
      RU1=RU1L+RU1S
      RV1=RV1L+RV1S
C
    1 CONTINUE
C
      CALL GETNMC
C
      IF (MONTH.LT.13) GO TO 12
```

```
C
C.... MONTH=13 IS ANNUAL AVERAGE CASE
C
      M1=13
      M2=13
      GO TO 13
C
C.... M1 IS FOR NORTHERN HEMISPHERE, M2 FOR SOUTHERN.  M2=M1+6
C     UNLESS M1=M2=13.
C
   12 M1=MONTH
      M2=MONTH + 6
C
C.... SOUTHERN HEMISPHERE DATA IS 6 MONTHS DISPLACED FOR GROVES,
C        STATIONARY PERTURBATIONS, AND RANDOM PERTURBATIONS
C
      IF (M2.GT.12) M2=M2 - 12
C
C.... READ GROVES PRESSURE DATA.  CONVERT TO REAL AND STORE IN ARRAY.
C
   13 CONTINUE
C
      CLOSE(25)
      FILNAM='NASPGROVES.F'
      OPEN(25,FILE=FILNAM,STATUS='OLD',FORM='FORMATTED',
     .     ERR=999,IOSTAT=IOS)
      REWIND(25)
C
      DO 100 I=1,234
      READ(25,111) IC,MI,IH,IY,IEX
  111 FORMAT(A2,13I4)
      IF (IC.NE.'P') GO TO 666
      IF (MI.EQ. M1) GO TO 30
      IF (MI.EQ. M2) GO TO 40
      GO TO 100
   30 KS=1
      GO TO 50
   40 KS=-1
   50 IH=(IH-20)/5
      TENX=10.**IEX
      DO 60 J=1,10
      K=10+KS*(J-1)
   60 PG(IH,K) = IX(J)*TENX
  100 CONTINUE
C
C.... READ GROVES DENSITY DATA.  CONVERT TO REAL AND STORE IN ARRAY.
C
      DO 200 I=1,234
      READ(25,111) IC,MI,IH,IX,IEX
      IF (IC.NE.'D') GO TO 666
      IF (MI.EQ. M1) GO TO 130
      IF (MI.EQ. M2) GO TO 140
      GO TO 200
  130 KS=1
      GO TO 150
  140 KS=-1
  150 IH=(IH-20)/5
      TENX=10.**IEX
      DO 160 J=1,10
      K=10+KS*(J-1)
  160 DG(IH,K) = IX(J)*TENX
  200 CONTINUE
C
C.... READ GROVES TEMPERATURE DATA.  CONVERT TO REAL AND STORE IN ARRAY.
```

```
C
      DO 300 I=1,234
      READ(25,111) IC,MI,IH,IX,IEX
      IF (IC .NE. 'T') GO TO 666
      IF (MI.EQ.M1) GO TO 230
      IF (MI.EQ.M2) GO TO 240
      GO TO 300
  230 KS=1
      GO TO 250
  240 KS=-1
  250 IH=(IH-20)/5
      TENX=10.**IEX
      DO 260 J=1,10
      K=10+KS*(J-1)
  260 TG(IH,K) = IX(J)*TENX
  300 CONTINUE
C
C.... ANNUAL MEAN CASE - BOTH HEMISPHERES EQUAL.
C
      IF (MONTH.LT.13) GO TO 308
C
      DO 304 I=1,18
      DO 304 J=1,9
      J20=20-J
      PG(I,J)=PG(I,J20)
      DG(I,J)=DG(I,J20)
      TG(I,J)=TG(I,J20)
  304 CONTINUE
C
C.... READ STATIONALY PERTURBATIONS DATA.  CONVERT TO REAL AND STORE
C     IN PSP, DSP, AND TSP ARRAYS.
C
  308 DO 360 I=1,1248
      READ(25,112) NDATA(1),(NDATA(KK),KK=2,19)
  112 FORMAT(A2,18I4)
      IC=NDATA(1)
      MI=NDATA(2)
      IH=NDATA(3)
      LON=NDATA(4)
      DO 311 K=1,5
      IP(K)=NDATA(4+K)
      ID(K)=NDATA(9+K)
  311 IT(K)=NDATA(14+K)
      IF (IC .NE. 'S' ) GO TO 666
      IF (MI.EQ.M1) GO TO 320
      IF (MI.EQ.M2) GO TO 330
      GO TO 360
  320 KS=1
      GO TO 340
  330 KS=-1
  340 ISH=2+(IH-44)/8
      L=(LON+20)/30
      IF(IH.LT.52) ISH = (IH-20)/10
      IF (IH.GT.84) ISH=8
      DO 350 J=1,5
      K=5+KS*(J+(KS-1)/2)
      PSP(ISH,K,L) = IP(J)/1000.
      DSP(ISH,K,L) = ID(J)/1000.
  350 TSP(ISH,K,L) = IT(J)/1000.
  360 CONTINUE
C
C.... ANNUAL MEAN CASE - BOTH HEMISPHERES EQUAL.
C
      IF (MONTH.LT.13) GO TO 369
      DO 364 I=1,8
      DO 364 K=1,12
      DO 364 J=1,5
      J10=11-J
      PSP(I,J,K)=PSP(I,J10,K)
      DSP(I,J,K)=DSP(I,J10,K)
      TSP(I,J,K)=TSP(I,J10,K)
  364 CONTINUE
C
  369 CONTINUE
C
C.... READ RANDOM PERTURBATIONS.
C
      CLOSE(25)
      FILNAM='NASPRRW.F    '
      OPEN(25,FILE=FILNAM,STATUS='OLD',FORM='FORMATTED',
     .     ERR=999,IOSTAT=IOS)
      REWIND(25)
C
      DO 430 I=1,260
      READ (25,112) IC,MI,IH,IP,ID,IT
  385 IF (IC .NE. 'R' ) GO TO 666
      IF (MI.EQ.M1) GO TO 390
      IF (MI.EQ.M2) GO TO 400
      GO TO 430
  390 KS=1
      GO TO 410
  400 KS=-1
  410 IF (IH.LT.95) IHR=(IH-20)/5
      IF (IH.GE.95) IHR = 14 + (IH - 80) / 20
      DO 420 J=1,5
      K = 5 + KS * (J + (KS - 1) / 2)
      PR(IHR,K) =(IP(J)*RPSCALE/1000.)**2
      DR(IHR,K) =(ID(J)*RPSCALE/1000.)**2
  420 TR(IHR,K) =(IT(J)*RPSCALE/1000.)**2
  430 CONTINUE
C
C.... ANNUAL MEAN CASE - BOTH HEMISPHERES EQUAL.
C
      IF (MONTH.LT.13) GO TO 460
C
      DO 435 I=1,20
      DO 435 J=1,5
      J10=11-J
      PR(I,J)=PR(I,J10)
      DR(I,J)=DR(I,J10)
      TR(I,J)=TR(I,J10)
  435 CONTINUE
C
C.... READ RANDOM WIND STANDARD DEVIATIONS.
C
  460 DO 490 I=1,325
      READ(25,111) IC,MI,IH,IP,ID
  467 IF (IC .NE. 'RW') GO TO 666
      IF (MI.EQ.M1) GO TO 470
      IF (MI.EQ.M2) GO TO 475
      GO TO 490
  470 KS=1
      GO TO 480
  475 KS=-1
  480 IF (IH.LT.95) IHP=1+IH/5
      IF (IH.GE.95) IHP=19+(IH-80)/20
      DO 485 J=1,5
      K=5+KS*(J+(KS-1)/2)
      UR(IHR,K)=(IP(J)*RPSCALE)**2
  485 VR(IHR,K)=(ID(J)*RPSCALE)**2
  490 CONTINUE
C
C.... ANNUAL MEAN CASE - BOTH HEMISPHERES EQUAL.
C
      IF (MONTH.LT.13) GO TO 500
      DO 495 I=1,25
      DO 495 J=1,5
      J10=11-J
      UR(I,J)=UR(I,J10)
      VR(I,J)=VR(I,J10)
  495 CONTINUE
C
  500 CONTINUE
C
C.... READ ANNUAL PRESSURE, DENSITY, AND TEMPERATURE PERCENTS.
C
      CLOSE(25)
      FILNAM='NASPPPWCS.F '
C
      OPEN(25,FILE=FILNAM,STATUS='OLD',FORM='FORMATTED',
     .     ERR=999,IOSTAT=IOS)
      REWIND(25)
C
      DO 840 I=1,25
      READ(25,112) IC,MI,IH,IP,ID,IT
  820 IF(IH.GT.90) IH=70+(IH/4)
      IH=1+(IH/5)
      IF (IC .NE.'P' .OR. IH .NE. I) GO TO 666
      DO 830 J=1,5
      PLP(I,J+5)=IP(J)/1000.
      PLP(I,6-J)=IP(J)/1000.
      DLP(I,J+5)=ID(J)/1000.
      DLP(I,6-J)=ID(J)/1000.
      TLP(I,J+5)=IT(J)/1000.
  830 TLP(I,6-J)=IT(J)/1000.
  840 CONTINUE
C
C.... READ WIND ANNUAL PERCENTS.
C
      DO 865 I=1,25
      READ(25,113) IC,MI,IH,IP,ID
  113 FORMAT(A2,12I5)
  855 IF(IH.GT.90) IH=70+(IH/4)
      IH=1+(IH/5)
      IF (I .NE. IH .OR. IC .NE. 'PW') GO TO 666
      DO 860 J=1,5
      ULP(I,J+5)=IP(J)/1000.
      ULP(I,6-J)=IP(J)/1000.
      VLP(I,J+5)=ID(J)/1000.
  860 VLP(I,6-J)=ID(J)/1000.
  865 CONTINUE
C
C.... READ SMALL SCALE VELOCITY PERTURBATION CORRELATIONS.
C
      DO 888 I=1,25
      READ(25,113) IC,MI,IH,IP,ID
  880 IF(IH.GT.90) IH=70+(IH/4)
      IH=1+(IH/5)
      IF (IH .NE. I .OR. IC .NE. 'DS') GO TO 666
      DO 885 J=1,5
      UDS(I,J+5)=(IP(J)/1000.)
      UDS(I,6-J)=(IP(J)/1000.)
      VDS(I,J+5)=(ID(J)/1000.)
  885 VDS(I,6-J)=(ID(J)/1000.)
  888 CONTINUE
C
C.... READ LARGE SCALE VELOCITY PERTURBATION CORRELATIONS.
C
      DO 898 I=1,25
      READ(25,113) IC,MI,IH,IP,ID
  894 IF(IH.GT.90) IH= 70+(IH/4)
      IH=1+(IH/5)
      IF (IH .NE. I .OR. IC .NE. 'CL')GO TO 666
      DO 896 J=1,5
      UDL(I,J+5)=(IP(J)/1000.)
      UDL(I,6-J)=(IP(J)/1000.)
      VDL(I,J+5)=(ID(J)/1000.)
  896 VDL(I,6-J)=(ID(J)/1000.)
  898 CONTINUE
C
C.... READ QUASI-BIENNIAL OSCILLATIONS (PRESSURE AMPLITUDE AND
C     PRESSURE PHASE - DAYS PAST JAN 0, 1966).
C
      CLOSE(25)
      FILNAM='NASPQBO.F   '
      OPEN(25,FILE=FILNAM,STATUS='OLD',FORM='FORMATTED',
     .     ERR=999,IOSTAT=IOS)
      REWIND(25)
C
      DO 530 I=1,16
      READ(25,111) IC,IH,IX
  527 IF (IC .NE. 'QP' ) GO TO 666
      IH = (IH-5)/5
      DO 530 J=1,5
      PAQ(IH,J) = IX(2*J-1)/1000.
  530 PDQ(IH,J) = IX(2*J)*1.
      DO 531 I = 1,5
      PAQ(1,I) = 0.
 531  CALL PHASE(PDQ(2,I),15.,PDQ(3,I),20.,PDQ(1,I),10.)
C
C.... READ QBO DENSITY AMPLITUDE AND PHASE.
C
      DO 540 I=1,16
      READ(25,111) IC,IH,IX
  537 IF (IC .NE. 'QD') GO TO 666
      IH=(IH-5)/5
      DO 540 J=1,5
      DAQ(IH,J) = IX(2*J-1)/1000.
  540 DDQ(IH,J)=IX(2*J)*1.
      DO 541 I = 1,5
      DAQ(1,I) = 0.
 541  CALL PHASE(DDQ(2,I),15.,DDQ(3,I),20.,DDQ(1,I),10.)
C
C.... READ QBO TEMPERATURE AMPLITUDE AND PHASE.
C
      DO 550 I=1,16
      READ(25,111) IC,IH,IX
  547 IF (IC .NE. 'QT') GO TO 666
      IH = (IH- 5)/5
      DO 550 J=1,5
      TAQ(IH,J) = IX(2*J-1)/1000.
  550 TDQ(IH,J) = IX(2*J)*1.
      DO 551 I = 1,5
      TAQ(1,I) = 0.
 551  CALL PHASE(TDQ(2,I),15.,TDQ(3,I),20.,TDQ(1,I),10.)
C
```

```
C.... READ EASTWARD QBO WIND AMPLITUDE AND PHASE.
C
      DO 560 I=1,16
      READ(25,111) IC,IH,IX
  557 IF (IC .NE. 'QU') GO TO 666
      IH=(IH- 5)/5
      DO 560 J=1,5
      UAQ(IH,J) = IX(2 * J - 1) / 10.
  560 UDQ(IH,J)=IX(2*J)*1.
      DO 561 I = 1,5
      UAQ(1,I) = 0.
 561  CALL PHASE(UDQ(2,I),15.,UDQ(3,I),20.,UDQ(1,I),10.)
C
C.... READ NORTHWARD QBO WIND AMPLITUDE AND PHASE.
C
      DO 570 I=1,16
      READ(25,111) IC,IH,IX
  567 IF (IC .NE. 'QV') GO TO 666
      IH=(IH- 5)/5
      DO 570 J=1,5
      VAQ(IH,J) = IX(2 * J - 1) / 10.
  570 VDQ(IH,J) = IX(2*J)*1.
      DO 571 I = 1,5
      VAQ(1,I) = 0.
 571  CALL PHASE(VDQ(2,I),15.,VDQ(3,I),20.,VDQ(1,I),10.)
C
C.... READ IN SPHERICAL HARMONICS COEFFICIENTS
C
      CLOSE(25)
      FILNAM='NASPSP.F    '
      OPEN(25,FILE=FILNAM,STATUS='OLD',FORM='FORMATTED',
     .     ERR=999,IOSTAT=IOS)
      REWIND(25)
C
      DO 615 IFR=1,MN
      DO 613 JFR=1,14
      READ(25 ,640) IF1,IF2,(IDUM(I),I=1,9)
  640 FORMAT(2X,11I6)
      DO 613 I=1,9
 613  UCOEF(JFR,I)=FLOAT(IDUM(I))/100.
      DO 612 JFR=1,14
      READ(25 ,640) IF1,IF2,(IDUM(I),I=1,9)
      DO 612 I=1,9
 612  VCOEF(JFR,I)=FLOAT(IDUM(I))/100.
 615  CONTINUE
C
      CLOSE(25)
C
 621  R=H1
      IF(H1.LT.25.) R=25.
      CALL RTERP(R ,PHI1,PR,DR,TR,SP1,SD1,ST1)
      CALL INTRUV(PLP,DLP,H1,PHI1,PLP1,DLP1)
      CALL INTRUV(TLP,DLP,H1,PHI1,TLP1,R)
C
      SP1L=SQRT(PLP1*ABS(SP1))*100.
      SP1S=SQRT((1.-PLP1)*ABS(SP1))*100.
      SD1L=SQRT(DLP1*ABS(SD1))*100.
      SD1S=SQRT((1.-DLP1)*ABS(SD1))*100.
      ST1L=SQRT(TLP1*ABS(ST1))*100.
      ST1S=SQRT((1.-TLP1)*ABS(ST1))*100.
C
      CALL INTRUV(UR,VR,H1,PHI1,SU1,SV1)
      CALL INTRUV(ULP,VLP,H1,PHI1,ULP1,VLP1)
C
```

```
      SU1L=SQRT(ULP1*ABS(SU1))
      SU1S=SQRT((1.-ULP1)*ABS(SU1))
      SV1L=SQRT(VLP1*ABS(SV1))
      SV1S=SQRT((1.-VLP1)*ABS(SV1))
C
      CALL INTRUV(UDL,VDL,H1,PHI1,UDL1,VDL1)
      CALL INTRUV(UDS,VDS,H1,PHI1,UDS1,VDS1)
C
      UDL1=UDL1*100.
      VDL1=VDL1*100.
      UDS1=UDS1*100.
      VDS1=VDS1*100.
C
      RP1L=RP1L/100.
      RD1L=RD1L/100.
      RT1L=RT1L/100.
      SP1L=SP1L/100.
      SD1L=SD1L/100.
      ST1L=ST1L/100.
      RP1S=RP1S/100.
      RD1S=RD1S/100.
      RT1S=RT1S/100.
      SP1S=SP1S/100.
      SD1S=SD1S/100.
      ST1S=ST1S/100.
      UDL1=UDL1/100.
      VDL1=VDL1/100.
      UDS1=UDS1/100.
      VDS1=VDS1/100.
      RETURN
C
  666 WRITE(6,700) FILNAM
  700 FORMAT('  ERROR IN SETUP INPUT FROM ',A12)
      STOP
C
  999 CONTINUE
C
C.... OPEN ERROR ENCOUNTERED.
C
      WRITE(6,677) FILNAM,IOS
  677 FORMAT(' OPEN ERROR ON FILE ',A12,' STATUS = ',I3)
      STOP
C
      END
      SUBROUTINE SPHERE(MN,IH,PHIR,THETR,US,VS)
      COMMON/CHIC/DUM(18),IWSYM,UCOEF(14,9),VCOEF(14,9)
      DIMENSION Z(9)
      COSPHI=COS(PHIR)
      CSTHET=COS(THETR)
      SINPHI=SIN(PHIR)
      SNTHET=SIN(THETR)
      Z(1)=1.
      Z(2)=SINPHI
      Z(3)=CSTHET*COSPHI
      Z(4)=SNTHET*COSPHI
      Z(5)=(3*(SINPHI**2)-1)/2.
      Z(6)=CSTHET*(3*COSPHI*SINPHI)
      Z(7)=SNTHET*(3*COSPHI*SINPHI)
      Z(8)=(2*(CSTHET)**2-1)*(3*(COSPHI)**2)
      Z(9)=(2*SNTHET*CSTHET)*(3*(COSPHI)**2)
    5 IH5=IH/5-4
      IFP=9
      IF(IH.GT.65)IFP=4
      US=0.
      VS=0.
      DO 10 I=1,IFR
      US=US+Z(I)*UCOEF(IH5,I)
      VS=VS+Z(I)*VCOEF(IH5,I)
   10 CONTINUE
      RETURN
      END
      SUBROUTINE STDATM(Z,T,P,D)
      DIMENSION ZS(49),TMS(49),WMS(49),PS(49)
      DATA (ZS(I),I=1,49)/0., 11.019, 20.063, 32.162, 47.35,
     * 51.413, 71.802, 86.000, 91., 94.,  97., 100., 103., 106.,
     * 108., 110., 112., 115., 120., 125., 130., 135., 140., 145.,
     * 150., 155., 160., 165., 170., 180., 190.,210.,230., 265., 300.,
     * 350., 400., 450., 500., 550., 600., 650.,700.,750.,800.,850.,
     * 900., 950.,1000./
      DATA (TMS(I),I=1,49)/288.15, 216.65,216.65, 228.65, 270.65,270.65,
     * 214.65, 186.95, 186.87,187.74,190.4,195.08,202.23,212.89,223.29,
     * 240.,264.,300.,360.,417.23,469.27,516.59,559.63,598.78,
     * 634.39,666.8,696.29,723.13,747.57,790.07,825.31,
     * 878.84,915.78,955.2,976.01,990.06,995.83,998.22,
     * 999.24,999.67,999.85,999.93,999.97,999.99,999.99,
     *1000.,1000.,1000.,1000./
      DATA (WMS(I),I=1,49)/28.9644, 28.9644, 28.9644, 28.9644, 28.9644,
     * 28.9644, 28.9644, 28.9522, 28.889,28.783,  28.62,28.395,28.104,
     *27.765,27.521,27.268,27.020,26.680,26.205,25.803,25.436,25.087,
     *24.749,24.422,24.103,23.792,23.488,23.192,22.902,
     *22.342,21.809,20.825,19.952,18.688,17.726,16.735,
     *15.984,15.247,14.330,13.092,11.505,9.718,7.998,
     *6.579,5.543,4.849,4.404,4.122,3.940/
      DATA (PS(I),I=1,49)/1013.25, 226.32, 54.7487, 8.68014, 1.10905,
     * .66938,  .039564, 3.7338E-3, 1.5381E-3,9.0560E-4,5.3571E-4,
     * 3.2011E-4,1.9742E-4,1.2454E-4,9.3188E-5,7.1042E-5,5.5547E-5,
     * 4.0096E-5,2.5382E-5,1.7354E-5,1.25054E-5,9.3568E-6,
     * 7.2028E-6,5.6691E-6,4.5422E-6,3.6930E-6,3.0395E-6,
     * 2.5278E-6,2.1210E-6,1.5271E-6,1.1266E-6,6.4756E-7,
     * 3.9276E-7,1.7874E-7,8.7704E-8,3.4498E-8,1.4518E-8,
     * 6.4468E-9,3.0236E-9,1.5137E-9,8.2130E-10,4.8365E-10,
     * 3.1908E-10,2.2599E-10,1.7036E-10,1.3415E-10,1.0873E-10,
     * 8.9816E-11,7.5138E-11/
      IF(Z.LT.0.) GO TO 81
      RO=6356.766
      GO=9.80665
      WMO=28.9644
      RS=8314.32
      ZM=Z*1000.
      ROM=RO*1000.
      IF(Z.GE.86.) GO TO 6
      DO 3 I=1,7
      IF(ZS(I).LE.Z.AND.Z.LT.ZS(I+1)) GO TO 5
    3 CONTINUE
    5 ZL=RO*ZS(I)/(RO+ZS(I))
      ZU=RO*ZS(I+1)/(RO+ZS(I+1))
      ZLM=ZL*1000.
      ZUM=ZU*1000.
      WM=WMO
      HT=(RO*Z)/(RO+Z
      HM=HT*1000.
      G=(TMS(I+1)-TMS(I))/(ZU-ZL)
      GM=G*.001
      IF(G.LT.0..OR.G.GT.0.) GO TO 12
      P=PS(I)*EXP(-(GO*WMO*(HM-ZLM))/(RS*TMS(I)))*100.
      GO TO 13
   12 P=PS(I)*((TMS(I)/(TMS(I)+G*(HT-ZL)))**((GO*WMO)/(RS*GM)))*100.
   13 T=TMS(I)+G*(HT-ZL)
```

```
      GO TO 25
    6 DO 7 I=8,48
      IF(ZS(I).LE.Z.AND.Z.LT.ZS(I+1)) GO TO 8
    7 CONTINUE
      I=48
      IF(Z.LE.1000.)GO TO 8
   81 T=0.
      P=0.
      D=0.
      RETURN
    8 IF(I.NE.8)GO TO 31
      T=TMS(9)
      GO TO 39
   31 IF(I.LT.16.OR.I.GE.19)GO TO 32
      T=240.+12.0*(Z-110.0)
      GO TO 39
   32 IF(I.GE.19)GO TO 33
      T=263.1905-76.3232*SQRT(1.-((Z-91.)/19.9429)**2)
      GO TO 39
   33 XI=(Z-120.)*(RO+120.)/(RO+Z)
      T=1000.-640.*EXP(-0.01875*XI)
   39 J=I
      IF(I.EQ.48)J=I-1
      Z0=ZS(J)
      Z1=ZS(J+1)
      Z2=ZS(J+2)
      WMA=WMS(J)*(Z-Z1)*(Z-Z2)/((Z0-Z1)*(Z0-Z2))+WMS(J+1)*(Z-Z0)
     &*(Z-Z2)/((Z1-Z0)*(Z1-Z2))+WMS(J+2)*(Z-Z0)*(Z-Z1)/
     &((Z2-Z0)*(Z2-Z1))
      ALP0=ALOG(PS(J))
      ALP1=ALOG(PS(J+1))
      ALP2=ALOG(PS(J+2))
      ALPA=ALP0*(Z-Z1)*(Z-Z2)/((Z0-Z1)*(Z0-Z2))+ALP1*(Z-Z0)
     &*(Z-Z2)/((Z1-Z0)*(Z1-Z2))+ALP2*(Z-Z0)*(Z-Z1)/
     &((Z2-Z0)*(Z2-Z1))
      ALPB=ALPA
      WMB=WMA
      IF(I.EQ.8.OR.I.EQ.48)GO TO 24
      J=J-1
      Z0=ZS(J)
      Z1=ZS(J+1)
      Z2=ZS(J+2)
      ALP0=ALOG(PS(J))
      ALP1=ALOG(PS(J+1))
      ALP2=ALOG(PS(J+2))
      ALPB=ALP0*(Z-Z1)*(Z-Z2)/((Z0-Z1)*(Z0-Z2))+ALP1*(Z-Z0)
     &*(Z-Z2)/((Z1-Z0)*(Z1-Z2))+ALP2*(Z-Z0)*(Z-Z1)/
     &((Z2-Z0)*(Z2-Z1))
      WMB=WMS(J)*(Z-Z1)*(Z-Z2)/((Z0-Z1)*(Z0-Z2))+WMS(J+1)*(Z-Z0)
     &*(Z-Z2)/((Z1-Z0)*(Z1-Z2))+WMS(J+2)*(Z-Z0)*(Z-Z1)/
     &((Z2-Z0)*(Z2-Z1))
   24 P=100.*EXP((ALPA-ALPB)/2.)
      WM=(WMA+WMB)/2.
   25 D=(WM*P)/(RS*T)
   26 RETURN
      END
      SUBROUTINE TINF
      COMMON/IOTEMP/IOTEM1,IOTEM2,IUG,IUN,DD,XMJD,PHI1,PHI,
     .              NSAME,PP1, RD1, RT1, SP1, SD1, ST1, RU1, RV1, SU1, SV1,
     $ MN, IDA, IYP, EL, PHI1P,THET1P,G,PI,H,PHIP,THETP,F10,F10B,GI,
     $ IHR,MIN,NMORE,DH,HL,VL,DZ,DUMMY2(25)
      COMMON/COMJAC/XLAT,XLONG,SDA,SHA,DY,R,TE,EM
C
C   SUROUTINE TINF CALCULATES THE EXOSPHERIC TEMPERATURE ACCORDING TO JA
```

```
C     SAO NO. 313 ,19[illegible].
C
C LIST
C     F10 = SOLAR RADIO NOISE FLUX (XE-22 WATTS/M**2)
C     F10B= 81-DAY AVERAGE F10
C     GI  = GEOMAGNETIC ACTIVITY INDEX,AP
C     LAT = GEOGRAPHIC LATITUDE AT PERIGEE   (IN RAD)
C     SDA = SOLAR DECLINATION ANGLE          (IN RAD)
C     SHA = SOLAR HOUR ANGLE
C     DY  = DAY (DAY NUMBER/TROPICAL YEAR)? 1
C     R = 0.31  (DIURNAL FACTOR)
C
C   CONSTANTS -- C=SOLAR ACTIVITY VARIATION. BETA,ETC. = DIURNAL VARIATI
C                D=GEOMAGNETIC VARIATION. E=SIMIANNUAL VARIATION.
C
      C1 = 383.0
      C2 =   3.32
      C3 =   1.80
C
      PI = 3.14159265
      CON = 0.01745329252
      BETA= -37.0*CON
      GAMMA= 43.0*CON
      P    =   6.0*CON
      XM   =   2.5
      XNN  =   3.0
C
      D1 =  28.0
      D2 =   0.03
      D3 =   1.0
      D4 = 100.0
      D5 =  -0.08
C
      E1 =   2.41
      E2 =   0.349
      E3 =   0.206
      E4 = 360.*CON
      E5 = 226.5*CON
      E6 = 720.*CON
      E7 = 247.6*CON
      E8 =   0.1145
      E9 =   0.5
      E10= E4
      E11= 342.3*CON
      E12=  2.16
C
C SOLAR ACTIVITY VARIATION
C
      TC = C1 + C2*F10B + C3*(F10 - F10B)
C
C DIURNAL VARIATION
C
      ETA    = 0.5*ABS(XLAT - SDA)
      THETA  = 0.5*ABS(XLAT + SDA)
      TAU    = SHA + BETA + P*SIN(SHA + GAMMA)
      TPI=2*PI
      IF(TAU) 210,230,230
  210 IF(TAU+PI) 220,250,250
  220 TAU=TAU+TPI
      GO TO 210
  230 IF(TAU-PI) 250,250,240
  240 TAU=TAU-TPI
      GO TO 230
  250 CONTINUE
```

```
      A1 =(SIN(THETA))**XM
      A2 =(COS(ETA))**XM
      A3 =(COS(TAU/2.))**XNN
      B1 = 1.0 + R*A1
      B2 =(A2-A1)/B1
      TV = B1*( 1. + R*B2*A3)
      TL = TC*TV
C
C GEOMAGNETIC VARIATION
C
      TG = D3*GI + D4*(1-EXP(D5*GI))
C
C SEMIANNUAL VARIATION
C
      G3 = 0.5*(1.0 + SIN(E10*DY +E11) )
      G3 = G3**E12
      TAU1 = DY + E8*(G3 - E9)
      G1 = E2 + E3*(SIN(E4*TAU1 + E5))
      G2 = SIN(E6*TAU1+ E7)
      TS = E1 + F10B*G1*G2
C
C EXOSPHERIC TEMPERATURE
C
      TE = TL + TG + TS
      RETURN
      END
      SUBROUTINE TME
      COMMON/COMJAC/XLAT,XLONG,SDA,SHA,DY,R,T,EM
      COMMON/IOTEMP/IOTEM1,IOTEM2,IUG,IUN,DD,XMJD,PHI1,PHI,
     .           NSAME.RP1. RD1, RT1, SP1, SD1. ST1, RU1, RV1, SU1, SV1,
     $ MN, IDA, IYR, H1, PHI1R,THET1R,G,RI,H,PHIR,THETR,F10,F10B,AP,
     $ IHR,MIN,NMORE,DX,HL,VL,DZ,DUMMY2(25)
C
C LIST
C INPUT
C    MN=MONTH.  IDA=DAY. IYR=HEAR. HR = HOUR.  MIN = MINUTE
C    XLAT = LATITUDE (INPUT-GEOCENTRIC LATITUDE.)
C    XLONG= LONGITUDE(INPUT-GEOCENTRIC LONGITUDE. OUTPUT  -180 TO + 180)
C OUTPUT
C    SDA = SOLAR DECLINATION ANGLE (IN RAD)
C    SHA = SOLAR HOUR ANGLE (IN RAD)
C    DD  = DAY NUMBER FROM 1JAN.
C    DY  = DD/TROPICAL YEAR
C
C
C SET CONSTANTS
C
      YEAR = 365.2422
      YR=IYR
    6 DY = DD/YEAR
   30 FMJD = XMJD - 2435839.
C
C  COMPUTE GREENWICH MEAN TIME IN MINUTES GMT
C
      XHR =IHR
      XMIN = MIN
      GMT = 60*XHR + XMIN
C
C  COMPUTE GREENWICH MEAN POSITION - GP (IN DEG)
C
      XJ = (XMJD - 2415020.0)/(36525.0)
      A1=99.6909833
      A2 = 36000.76854
      A3 = 0.00038708
      A4 = 0.25068447
      GP = A1 + A2*XJ - A3*XJ*XJ + A4*GMT
      N = GP/360.
      XN = N
      GP = GP - XN*360.
C
C COMPUTE RIGHT ASCENSION POINT - RAP (IN DEG)
C
C     1ST CONVERT GEOCENTRIC LONGITUDE TO DEG LONGITUDE - WEST NEG $ EAS
C
      IFACT = XLONG/180.
      XFACT = IFACT
      XLONG = 360. * XFACT - XLONG
C
      RAP = GP + XLONG
      N = RAP/360.
      XN = N
      RAP = RAP - XN*360.
C
C COMPUTE CELESTIAL LONGITUDE - XLS (IN RAD) - -PI/2 TO +PI/2
C
      B1 = 0.017203
      B2 = 0.0335
      B3 = 1.410
      Y1 = B1*FMJD
      XLS = Y1 + B2*SIN(Y1) - B3
      TPI = 6.28318
      N = XLS/TPI
      XN = N
      XLS = XLS - XN*TPI
C
C COMPUTE SOLAR DECLINATION ANGLE - SDA (IN RAD)
C
      B4 = (TPI/360.)*23.45
      SDA = ASIN(SIN(XLS)*SIN(B4))
C
C COMPUTE RIGHT ASCENSION OF SUN - RAS (IN RAD) - -PI/2 TO +PI/2
C
      RAS = ASIN(TAN(SDA)/TAN(B4))
C
C PUT RAS IN SAME QUADRANT AS XLS
C
      PI = 3.14159265
      PI2 = PI/2.
      PI32= 3.*PI2
      RAS = ABS(RAS)
      TEMP = ABS(XLS)
      IF(TEMP - PI2) 130,130,100
  100 IF(TEMP - PI) 105,105,110
  105 RAS = PI - RAS
      GO TO 130
  110 IF(TEMP - PI32) 115,115,120
  115 RAS = PI + RAS
      GO TO 130
  120 RAS = TPI - RAS
  130 IF (XLS) 135,140,140
  135 RAS = -RAS
  140 CONTINUE
C
C COMPUTE SOLAR HOUR ANGLE - SHA (IN DEG) - -
C
      SHA = RAP*(PI/180.) - RAS
      IF(SHA) 210,230,230
  210 IF(SHA+PI) 220,230,250
  220 SHA=SHA+TPI
      GO TO 210
  230 IF(SHA-PI) 250,250,240
  240 SHA=SHA-TPI
      GO TO 230
  250 CONTINUE
C
      RETURN
      END
      SUBROUTINE USGRID
C
C.... THIS ROUTINE GENERATES THE DATA FOR THE 16 POINT GRID USED BY
C     THE GRAM PROGRAM AT ALTITUDES BELOW 25 KM (ALSO USED BETWEEN
C     25 AND 30 KM FOR INTERPOLATION WITH THE GROVES MODEL).
C
C.... THE DATA CONSISTS OF PRESSURE, DENSITY, TEMPERATURE,
C     PRESSURE VARIANCE, DENSITY VARIANCE, AND TEMPERATURE VARIANCE
C     FOR LATITUDES 20-65 AND LONGITUDES 35-140 WEST (CONTINENTAL US +).
C     DATA IS AT EACH 5 DEGREES OF LAT AND LONG, AND 0 TO 25 KM IN ONE
C     KM INCREMENT.
C
C.... ALTHOUGH NOT THE MOST EFFICIENT, THIS ROUTINE IS DESIGNED TO
C     INGERFACE WITH THE GRAM PROGRAM WITH MINIMUM OF IMPACT.  THUS THE
C     NEW GRID POINTS ARE DETERMINED AS BEFORE (AS A FUNCTION OF WHERE
C     YOU ARE AND WHERE YOU'RE HEADED).  THIS ROUTINE THEN OBTAINS THE
C     DATA FROM MEMORY AND TRANSFERS IT TO THE ARRAY LOCATIONS EXPECTED
C     BY THE GRAM PROGRAM.
C
C.... THIS ROUTINE ASSUMES THAT THE TRAJECTORY WHEN BELOW 25KM WILL
C     ALWAYS LIE WITHIN LAT 20-65 AND LONG 35-140 (WEST).  OUTSIDE THIS
C     AREA, DATA ON THE BORDER OF THE REGION WILL BE USED.  THUS THE
C     TRAJECTORY IS ASSUMED TO ALWAYS LIE WITHIN THE NATIONAL
C     METEOROLOGICAL CENTER DATA.  LOGIC FOR POLAR AND SOUTHERN
C     HEMISPHERE DATA HAS BEEN REMOVED.
C
C.... WRITTEN   23 JAN 89   L SCHILLING   NASA/ADFRF.
C
      COMMON /C4    /
     .                GLAT(16),GLON(16),NG,P(16,26),D(16,26),T(16,26),
     .                SP(16,26),SD(16,26),ST(16,26),PLON,CLON,HS
      COMMON /IOTEMP/ IOTEM1,IOTEM2,IUG   ,IUN   ,DDD   ,XMJD  ,PLAT  ,
     .                CLAT  ,NSAME ,RP1   ,RD1   ,RT1   ,SP1   ,SD1   ,
     .                ST1   ,RU1   ,RV1   ,SU1   ,SV1   ,MN    ,IDA   ,
     .                IYR   ,H1    ,PHI1R ,THET1R,GZ    ,RI    ,Z     ,
     .                PHIR  ,THETR ,F10   ,F10B  ,AP    ,IHR   ,MIN   ,
     .                NMORE ,DX    ,HL    ,VL    ,DZ    ,B     ,EPS   ,
     .                IOPP  ,LOOK  ,DUMMY(21)
      COMMON /PDTCOM/
     .                IU4   ,MONTH ,IOPR  ,PG(18,19),TG(18,19),
     .                DG(18,19),PSP(8,10,12),DSP(8,10,12),TSP(8,10,12),
     .                PAQ(17,5),DAQ(17,5),TAQ(17,5),PDQ(17,5),DDQ(17,5),
     .                TDQ(17,5),PR(20,10),DR(20,10),TR(20,10),UAQ(17,5),
     .                VAQ(17,5),UDQ(17,5),VDQ(17,5),UR(25,10),VR(25,10),
     .                PQ    ,DQ    ,TQ    ,UQ    ,VQ    ,PQA   ,DQA   ,
     .                TQA   ,UA    ,VA    ,IOPQ  ,PLP(25,10),DLP(25,10),
     .                TLP(25,10),ULP(25,10),VLP(25,10),UDL(25,10),
     .                VDL(25,10),UDS(25,10),VDS(25,10)
      COMMON /IPRTP/  IPRT
      COMMON /NASPGM/ PDAT(5720) ,DDAT(5720) ,TDAT(5720) ,
     .                        SPDAT(5720),SDDAT(5720),STDAT(5720)
C
      IF(NSAME.EQ.1) RETURN
C
      IPRT=0
      LOOK=0
      F = 0.017453293
      NG = 16
C
C.... LONG/LAT DISPLACEMENT FROM PREVIOUS TO CURRENT POSITION.
C
      DX = PLON - CLON
      DY = CLAT - PLAT
C
      IF (DY) 20,10,20
   10 IF (DX) 15,12,15
   12 K = 0
      GO TO 40
   15 THETA = 180. + SIGN(90.,DX)
      GO TO 30
   20 THETA = ATAN(DX/DY)/F
      IF (DY.GT.0.) THETA = THETA + 180.
      IF (THETA.LT.0.) THETA = THETA + 360.
C
C.... THETA = AZIMUTH ANGLE OF TRAJECTORY, USED TO ORIENT LAT-LON GRID
C     COMPUTE INDEX USED IN COMPUTED GO TO FOR 110 THRU 180
C
   30 K = INT((THETA + 67.5)/45.)
      IF (K.GT.8) K=K-8
C
C.... INITIAL ESTIMATE OF REFERNCE LATITUDE (LOWER LEFT GRID POINT)
C
   40 LAT0 = 5*INT(CLAT/5.)
      IF (CLAT.LT.0.) LAT0 = LAT0 - 5
C
C.... INITIAL ESTIMATE OF REFERENCE LONGITUDE (LOWER LEFT GRID POINT)
C
      LON0=5*INT(CLON/5.)
C
C.... ADJUSTS LAT0,LON0 ACCORDING TO DIRECTION OF TRAJECTORY AZIMUTH
C
      IF (K.GT.0) GO TO 100
      LAT0 = LAT0 - 5
      LON0= LON0 + 10
      GO TO 190
  100 GO TO (110,120,130,140,150,160,170,180),K
  110 LAT0 = LAT0-10
      LON0 = LON0 + 10
      GO TO 190
  120 LAT0 = LAT0-10
      LON0 = LON0+15
      GO TO 190
  130 LAT0 = LAT0-5
      LON0 = LON0+15
      GO TO 190
  140 LON0 = LON0+15
      GO TO 190
  150 LON0 = LON0+10
      GO TO 190
  160 LON0 = LON0+5
      GO TO 190
  170 LAT0 = LAT0-5
      LON0 = LON0+5
      GO TO 190
  180 LAT0 = LAT0-10
      LON0 = LON0+5
  190 IF (LON0.GE.360) LON0 = LON0 - 360
      IF (LAT0.GT.75) LAT0 = 75
      DLI=1.25
      IF(ABS(CLAT).GE.18) GO TO 192
C
      DLI=3.0
      LATO=-18
C
C.... LATITUDE, LONGITUDE GRID AT 5 DEGREE INTERVALS
C
  192 DO 195 I=1,4
      I12 = I+12
      DO 195 J=I,I12,4
      GLAT(J) = LATO + DLI*(J-I)
  195 GLON(J) = LONO - 5. * (I - 1)
C
C.... PUT DATA FROM STORED US GRID IN ARRAYS.  THIS DATA HAS ALREADY
C     BEEN TESTED FOR ZEROS, AND HAS GONE THROUGH ROUTINE 'ADJUST'
C     PRIOR TO STORAGE, THUS THIS PROCESS HAS BEEN REMOVED FROM THIS
C     ROUTINE.
C
C
C.... TRANSFER DATA TO GRAM PROGRAM ARRAYS.
C
      DO II=1,16
C
C....    COMPUTE SINGLY DIMENSIONED ARRAY INDEX.  LIMIT LATITUDE
C        AND LONGITUDE COMPONENTS TO STAY WITHIN TABLE BOUNDARIES.
C
         ILAT=GLAT(II)+0.1
         IF(ILAT.LT.20) ILAT=20
         IF(ILAT.GT.65) ILAT=65
C
         ILON=GLON(II)+0.1
         IF(ILON.LT. 35) ILON= 35
         IF(ILON.GT.140) ILON=140
C
         I1 = ((ILAT-20)/5)*572 + ((ILON-35)/5)*26 + 1
         I2 = I1 + 25
         I3 = 0
C
         DO I=I1,I2
            I3 = I3+1
            P (II,I3) =  PDAT(I)
            D (II,I3) =  DDAT(I)
            T (II,I3) =  TDAT(I)
            SP(II,I3) = SPDAT(I)
            SD(II,I3) = SDDAT(I)
            ST(II,I3) = STDAT(I)
         ENDDO
C
      ENDDO
C
      RETURN
      END
      SUBROUTINE WIND
      COMMON /WINCOM/PHO,FCORY,DX5,DY5,PX,PY,PXX,PXY,PYY,U,V,  T,TX,TY,
     $   DU,DV,P,UPRE,VPRE,DUPRE,DVPRE
      COMMON /IOTEMP/DUM1(7),PHI,DUM2(11),MN,DM2A(5),G,R,H,PHIR,
     $THETR,DUM3(15),FLAT,DUMMY(19)
      COMMON/CHIC/DUM(19),IWSYM,UCOEF(14,9),VCOEF(14,9)
      ABSPHI=ABS(PHIR)
      IF (ABSPHI.LT.0.017453293*FLAT) GO TO 40
      IF (PHO.GT.0. .AND. T.GT.0. .AND. ABS(FCORY).GT.0.) GO TO 20
      U = 0.
      V = 0.
      DU = 0.0
      DV = 0.0
      IF (ABS(FCORY).LE.0.) GO TO 31
      RETURN
   20 FCORX = FCORY*DX5/DY5
      U = - PY/(FCORY*RHO)
      V =   PX/(FCORX*RHO)
      DU = -(G*TY)/(FCORY*T)
      DV =   (G*TX)/(FCORX*T)
   31 IF(H.GT.20.AND.H.LT.95.)GOTO 99
      IF (ABSPHI.GE.0.017453293*FLAT) RETURN
   40 CONTINUE
      U=UPRE
      V=VPRE
      DU=DUPRE
      DV=DVPRE
      IF (H.GT.20. .AND. H.LT.95.) GO TO 99
      RETURN
C...SPHERICAL HARMONICS SECTION.......................................
   99 IH=INT(H)
      IF(IH.LT.25)GOTO 130
      IF(IH.GE.90)GOTO 140
      IH1=5*INT(H/5.)
      IH2=IH1+5
      CALL SPHERE(MN,IH1,PHIR,THETR,US,VS)
      CALL SPHERE(MN,IH2,PHIR,THETR,US2,VS2)
      FACS=(H-IH1)/5.
      U=US+(US2-US)*FACS
      V=VS+(VS2-VS)*FACS
      DU=(US2-US)/5000.
      DV=(VS2-VS)/5000.
      RETURN
C...LOW ALTITUDE FAIRING
  130 CALL SPHERE(MN,25,PHIR,THETR,US,VS)
      FACS=(H-20.)/5.
      FACG=1.-FACS
      U=FACG*U+FACS*US
      V=FACG*V+FACS*VS
      CALL SPHERE(MN,30,PHIR,THETR,US2,VS2)
      DUS=(US2-US)/5000.
      DVS=(VS2-VS)/5000.
      DU = FACG*DU + FACS*DUS
      DV = FACG*DV + FACS*DVS
      RETURN
C...HIGH ALTITUDE FAIRING
  140 CALL SPHERE(MN,90,PHIR,THETR,US,VS)
      FACS=(H-90.)/5.
      FACG=1.-FACS
      U=FACS*U+FACG*US
      V=FACS*V+FACG*VS
      CALL SPHERE(MN,85,PHIR,THETR,US2,VS2)
      DUS=(US-US2)/5000.
      DVS=(VS-VS2)/5000.
      DU=FACG*DU+FACS*DUS
      DV=FACG*DV+FACS*DVS
      RETURN
      END
!T72+
      FUNCTION RANF(X)                                                  1094.600
C                                                                       1094.700
C.... THIS FUNCTION IS A MODIFIED VERSION OF SUBROUTINE                 1094.800
C     RANDU, WHICH WAS WRITTEN FOR AN IBM 360.                          1094.900
C                                                                       1095.000
C.... DESCRIPTION OF VARIABLES:                                         1095.100
C        IX  - FOR THE FIRST ENTRY THIS MUST CONTAIN ANY ODD INTEGER    1095.200
```

```
C              NUMBER WITH NINE OR LESS DIGITS.  AFTER THE FIRST ENTRY, 1095.300
C              IX WILL BE SET TO THE PREVIOUS VALUE OF IY, COMPUTED BY  1095.400
C              THIS ROUTINE.                                            1095.500
C        IY  - A RESULTANT INTEGER RANDOM NUMBER REQUIRED FOR THE NEXT  1095.600
C              ENTRY TO THIS ROUTINE.  THE RANGE OF THIS NUMBER IS      1095.700
C              BETWEEN 0 AND 2**31                                      1095.800
C        YFL - THE RESULTANT UNIFORM DISTRIBUTED, FLOATING POINT, RANDOM1095.900
C              NUMBER IN THE RANGE 0.0 TO 1.0                           1096.000
C        X   - DUMMY ARGUMENT.                                          1096.100
C                                                                       1096.200
C.... REMARKS:                                                          1096.300
C        THIS ROUTINE IS SPECIFIC TO SYSTEM/360 AND WILL PRODUCE 2**29  1096.400
C        TERMS BEFORE REPEATING.                                        1096.500
C                                                                       1096.600
C.... METHOD:                                                           1096.700
C        POWER RESIDUE METHOD DISCUSSED IN IBM MANUAL C20-8011,         1096.800
C        RANDOM NUMBER GENERATION AND TESTING.                          1096.900
C                                                                       1097.000
      DATA IX /113/                                                     1097.100
C                                                                       1097.200
      IY = IX*65539                                                     1097.300
C                                                                       1097.400
      IF(IY) 5,6,6                                                      1097.500
C                                                                       1097.600
    5 IY = IY + 2147483647 + 1                                          1097.700
C                                                                       1097.800
    6 YFL = IY                                                          1097.900
C                                                                       1098.000
      RANF = YFL*0.4656613E-9                                           1098.100
C                                                                       1098.200
      IX = IY                                                           1098.300
C                                                                       1098.400
      RETURN                                                            1098.500
      END                                                               1098.600
!!T72-
C
      SUBROUTINE GRNGE ( RIO, RI, AE, BE, RES, IEMO, ICOORD, GRNGE )
C
C=====================================================================C
C                                                                     C
C     PURPOSE   1) COMPUTE THE GROUND RANGE FROM POSITION (RIO) TO    C
C                  POSITION (RI) IN THE APPROPRIATE COORDINATE SYSTEM C
C                  AS GIVEN BY (ICOORD) AND BY THE EARTH MODEL FLAG   C
C                  (IEMO)                                             C
C                                                                     C
C     INPUTS    DESCRIPTION                                           C
C                                                                     C
C     (RIO)     MISSILE INITIAL POSITION VECTOR (RXI,RYI,RZI) (M)     C
C     (RI)      MISSILE CURRENT POSITION VECTOR (RXI,RYI,RZI) (M)     C
C     (AE)      EARTH MODEL SEMI-MAJOR AXIS (M)                       C
C     (BE)      EARTH MODEL SEMI-MINOR AXIS (M)                       C
C     (RES)     SPHERICAL EARTH RADIUS (M)                            C
C     (IEMO)    EARTH MODEL FLAG:                                     C
C                 = 0 - FLAT EARTH                                    C
C                   1 - FLAT EARTH                                    C
C                   2 - SPHERICAL EARTH                               C
C                   3 - OBLATE EARTH                                  C
C     (ICOORD)  COORDINATE SYSTEM OF EARTH-MODEL FLAG:                C
C                 = 1 - FLAT EARTH OR SPHERICAL EARTH - AT SURFACE    C
C                           X - LOCAL EAST                            C
C                           Y - LOCAL NORTH                           C
C                           Z - UP                                    C
C                   2 - OBLATE OR SPHERICAL EARTH - EARTH CENTERED    C
C                           X - B.R.P. MERIDIAN                       C
```

```
C                         Y - NORMAL - EQUATORIAL PLANE                   C
C                         Z - THROUGH NORTH POLE                          C
C                                                                         C
C     OUTPUTS   DESCRIPTION                                               C
C                                                                         C
C     (GRNGE)   GROUND RANGE (ALONG THE SURFACE OF THE EARTH MODEL)       C
C               FROM POSITION (RIO) TO POSITION (RI)                      C
C                                                                         C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - C
C                                                                         C
C     REQUIREMENTS  1  SINGLE TO DOUBLE PRECISION          - DBLE         C
C                   2  DOUBLE TO SINGLE PRECISION          - SNGL         C
C                   3  D.P. SQUARE ROOT FUNCTION           - DSQRT        C
C                   4  D.P. ARCTANGENT FUNCTION (+/- PI)   - DATAN2       C
C                   5  D.P. SINE FUNCTION                  - DSIN         C
C                   6  D.P. ARCCOSINE FUNCTION             - DACOS        C
C                                                                         C
C - - - - - - -   - - - - - - - - - - - - - - - - - - - - - - - - - - - - C
C                                                                         C
C     RESTRICTIONS 1) THE FLAGS FOR EARTH MODEL AND COORDINATE SYSTEM     C
C                     MUST CONFORM TO THE FOLLOWING:                      C
C                         IEMO    ICOORD                                  C
C                        ------   ------                                  C
C                           0        1                                    C
C                           1        1                                    C
C                           2        1                                    C
C                           2        2                                    C
C                           3        2                                    C
C                                                                         C
C=========================================================================C
C
      DIMENSION  RI(3   RIO(3)
C
      DOUBLE PRECISION  DRXI, DRYI, DRZI, DRXE, DRYE, DRZE
      DOUBLE PRECISION  DXI2, DYI2, DZI2, DRE,  DECC
      DOUBLE PRECISION  DXE2, DYE2, DZE2, DAE,  DBE,  DTHE
      DOUBLE PRECISION  DRMI, DRME, DRR2, DTHR, DRES
      DOUBLE PRECISION  DSLA, DALG, DALA, DSL2, DRM
C
      DOUBLE PRECISION  DF0,  DF1,  DF2
C
      DATA  DF0, DF1, DF2 /  0.0D0, 1.D0, 2.0D0 /
C
      IF( IEMO .GT. 1 ) GO TO 10
C
Cswswswswswswswswswswswswswswswswswswswswswswswswswswswswswswswswswswsws
C
C....... FLAT EARTH       IEMO = 0 OR 1,  ICOORD = 1 (N/A)
C        ==========
C
      GRNGE = SQRT( (RIO(1) - RI(1))**2 + (RIO(2) - RI(2))**2 )
C
      GO TO 100
C
Cswswswswswswswswswsws swswswswswswswswswswswswswswswswswswswswswswsws
C
   10 CONTINUE
C
C....... OTHER THAN FLAT EARTH
C                  ==========
C
      DRXI = DBLE (  RI(1) )
      DRYI = DBLE (  RI(2) )
      DRZI = DBLE (  RI(3) )
```

```
C
      DRXE = DBLE ( RIO(1) )
      DRYE = DBLE ( RIO(2) )
      DRZE = DBLE ( RIO(3) )
C
      IF( IEMO .EQ. 3 ) GO TO 20
C
C....... SPHERICAL EARTH      IEMO = 2,  ICOORD = 1 OR 2
C         =========
C
      DRES = DBLE ( RES )
C
      IF( ICOORD .EQ. 1 ) DRZI = DRZI + DRES
      IF( ICOORD .EQ. 1 ) DRZE = DRZE + DRES
C
      GO TO 40
C
Cswswswswswswswswswswswswswswswswswswswswswswswswswswswswswswswswswsws
C
   20 CONTINUE
C
C....... OBLATE EARTH      IEMO = 3,  ICOORD = 2
C         ======
C
      DAE  = DBLE ( AE )
      DBE  = DBLE ( BE )
      DECC = (DAE/DBE)**2
C
      DXI2 = DRXI**2
      DYI2 = DRYI**2
C
      DALG = DSQRT ( DXI2  + DYI2 )
      DALA = DF0
      IF(( DRZI .NE. DF0 ).OR.
     *   ( DALG .NE. DF0 ))   DALA = DATAN2( DRZI, DALG )
C
      DSLA  = DSIN( DALA )
      DSL2  = DSLA*DSLA
C
      DRE   = DAE/DSQRT( DF1 + (DECC-DF1)*DSL2 )
C
      DXE2 = DRXE**2
      DYE2 = DRYE**2
C
      DALG = DSQRT ( DXE2  + DYE2 )
      DALA = DF0
      IF(( DRZE .NE. DF0 ).OR.
     *   ( DALG .NE. DF0 ))   DALA = DATAN2( DRZE, DALG )
C
      DSLA = DSIN( DALA )
      DSL2 = DSLA*DSLA
C
      DRES = ( DRE + DAE/DSQRT( DF1 + (DECC-DF1)*DSL2 ) )/DF2
C
Cswswswswswswswswswswswswswswswswswswswswswswswswswswswswswswsws
C
   40 CONTINUE
C
C....... GROUND RANGE COMPUTATION
C
      DRMI = DRXI*DRXI + DRYI*DRYI + DRZI*DRZI
      DRME = DRXE*DRXE + DRYE*DRYE + DRZE*DRZE
C
      DTHR = DRXI*DRXE + DRYI*DRYE + DRZI*DRZE
      DRR2 = DSQRT (   -MI*DRME )
      DTHE = DF0
C
      IF( DRR2 .NE.    0 ) DTHE = DTHR/DRR2
      IF( DTHE .GT.    1 ) DTHE =  DF1
      IF( DTHE .LT.   -1 ) DTHE = -DF1
C
      DTHE = DACOS(   HE )
      DRM  = DRES*DT
C
      GRNGE = SNGL (   M )
C
Cswswswswswswswsws   swswswswswswswswswswswswswswswswswsws
C
  100 CONTINUE
C
      RETURN
      END
!!T72+
      SUBROUTINE GET                                                    GETN   1
C                                                                       GETN   2
C     READS  SETUP  DATA TAPE, OR NMC GRID DATA CARDS,                  GETN   3
C     AND WRITES SCRATCHE FILE FOR USE BY SELEC4.                       GETN   4
C                                                                       GETN   5
      DIMENSION IP(15                                                   GETN   6
C                                                                       GETN   7
      COMMON /IOTEMP  IOTEM1,IOTEM2,IUG,IUN,IDUM(60)                    GETN   8
C                                                                       GETN   9
      OPEN(UNIT=IOTEM .STATUS='SCRATCH',FORM='UNFORMATTED')             GETN  10
      NREC=0                                                            GETN  11
      IF(IUN.EQ.5) GO TO 2                                              GETN  12
C
      OPEN(UNIT=IUN.F LE='NMC.DAT',STATUS='OLD',READONLY)               GETN  13
1     READ(IUN,300.EN =90) N,IP                                         GETN  14
300   FORMAT(A2,19I7                                                    GETN  15
      IF(N.NE.'N  ') G  TO 6                                            GETN  16
      GO TO 3                                                           GETN  17
    2 READ(5,100) IP                                                    GETN  18
  100 FORMAT(15I5)                                                      GETN  19
    3 DO 4 I=1,15,3                                                     GETN  20
      M=IP(I)                                                           GETN  21
      IF(M.LT.1) GO T  5                                                GETN  22
      IJ=IP(I+1)*1000  P(I+2)                                           GETN  23
      WRITE(IOTEM2) I                                                   GETN  24
      NREC=NREC+1                                                       GETN  25
    4 CONTINUE                                                          GETN  26
      IF(IUN.EQ.5) GO TO 2                                              GETN  27
      GO TO 1                                                           GETN  28
    5 IF(NREC.NE.19   GO TO 6
      IF(IUN.EQ.5) G  O 42
      IF(IUN.NE.IUG)   TO 42                                            GETN  29
C     MOVES PAST FIR  EOF ON UNIT IUG                                   GETN  30
41    READ(IUG,9999,E  =42) IDUMMY                                      GETN  31
9999  FORMAT(A10)                                                       GETN  32
      GO TO 41                                                          GETN  33
42    RETURN                                                            GETN  36
 6    STOP                                                              GETN  37
90    WRITE(6,400) IU
  400 FORMAT (1H , PREMATURE END-OF-FILE FOUND ON UNIT ',I2,
     $'0 CALLED FROM SUBROUTINE GETNMC.')                               GETN  40
      STOP                                                              GETN  41
      END
!!T72-
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-------------------------------------------------------------------------------
      subroutine SaveMissionFile
c-------------------------------------------------------------------------------
c     Save a mission file's data and resource forks using a newly opened
c     file (which may overwrite an existing file).  Then, close the file.
c     (The dialog box contents have already been checked for validity.)
c

c.....file info

      include 'FileInfo.inc'
      include 'RunSetup.inc'
      include 'traj.inc'
      include 'TrjCom.inc'

c.....set up pointer for QuickDraw globals

      pointer / QDGlobals /       qdg
      common  / QDGPtr  /          qdg

c.....various character strings

      string*255                  Prompt
      string*255                  MissionFileName
      string*255                  RezLabel
      string*255                  ItemText
      character*255               CharString

c.....reply record

      record / SFReply    /       SFR

c.....prompt location

      record / Point      /       PrmptPt

c.....cursor handle

      record / CursHandle /       CursorHndl

c.....file information parameters

      record /  SFTypeList / SFT
      record / FInfo       / fndrInfo
      character*4            FilTyp
      character*4            fMaker
      string*255             FilNam

c.....I/O error flags

      integer*2              iosErr,      ios

c.....screen position info

      integer*2 menuHeight
      integer*4 left, bottom, top, right
      integer*4 dialo.Height, dialogWidth

c.....dimensions of st andard SFPutFile dialog box (InsideMac, Chapter 47)

      parameter           ( dialogHeight       =  136 )
      parameter           ( dialogWidth        =  348 )

c.....handle for STP  ata

      record / String andle / STRHndl
      record / Handle       / RezHndl

c.....varia  e name l  els for external plot file

      character*16 Na  s(7)

c------------------- --------------------------------------------------------

c.....set the prompt

      prompt = 'Save  ission data file as'

c.....get the menuBe  t (don't assume it is fixed at 20!)

      menuHeight = G  MBarHeight()

c.....get the screen  tents (use i*4 for screen math per Mac Tech Note 117)

      left   = qdg^.s  eenBits.bounds.left
      right  = qdg^.s  eenBits.bounds.right
      bottom = qdg^.s  eenBits.bounds.bottom
      top    = qdg^.s  eenBits.bounds.top + int4(menuHeight)

c.....set the left an  top edges of the save file dialog

      PrmptPt.H =    right - left) - dialogWidth  ) / 2
      PrmptPt.V = (   bottom - top) - dialogHeight ) / 2 ) + menuHeight

c.....check to see if  e already have the filename for saving

      if( .not. iGot  File ) then
        MissionFileNa    = 'BDPS Mission Data'
      endif

c.....get the target  lename for the save (put) operation

      call SFPutFile    %val(PrmptPt), %val(Prompt), %val(MissionFileName),
     &                  %val(int4(nil)), %ref(SFR) )

c.....RETURN if no mi sion data file was selected (cancel)

      if ( .not. SFR. ood ) then
        return
      else

        MissionFileN     = SFR.fName

        iosErr = Set    %val(int4(nil)) , %val(SFR.vRefNum) )

c.......open the new  te fork

        open( unit=1   file=MissionFileName, creator='MDoF', filetype='rtdf',
     &        status= ew', access='sequential' )
```

```
c.......open the ne    source fork

        call CreateR   ile( %ref(SFR.fname) )
        ios    = Res   or
        RefNum = Ope   sFile ( %ref(SFR.fname) )
        ios    = Res.  or

c.......setup the mis  on label resource

        STRHndl  = N   andle( %val(int4(255) ) )
        RezLabel = 'M  sion Text'
        call AddReso   e( %val(STRHndl), %val('STR '),
     &                    %val(int2(rOldMissionText)), %ref(RezLabel) )
        RezHndl = STR  dl
        call WriteRes  rce( %val(RezHndl) )

c.......setup the lat  ude resource

        STRHndl  = Ne. andle( %val(int4(255) ) )
        RezLabel = 'In tial Latitude'
        call AddResour e( %val(STRHndl), %val('STR '),
     &                    %val(int2(rOldLatitude)), %ref(RezLabel) )
        RezHndl = STR ndl
        call WriteRes  rce( %val(RezHndl) )

c.......setup the lon  tude resource

        STRHndl  = Ne  andle( %val(int4(255) ) )
        RezLabel =  In tial Longitude'
        call AddResour e( %val(STRHndl), %val('STR '),
     &                    %val(int2(rOldLongitude)), %ref(RezLabel) )
        RezHndl = STRHndl
        call WriteResource( %val(RezHndl) )

c.......setup the duration resource

        STRHndl  = NewHandle( %val(int4(255) ) )
        RezLabel = 'Flight Duration'
        call AddResource( %val(STRHndl), %val('STR '),
     &                    %val(int2(rOldDuration)), %ref(RezLabel) )
        RezHndl = STRHndl
        call WriteRes urce( %val(RezHndl) )

c.......setup the altitude resource

        STRHndl  = NewHandle( %val(int4(255) ) )
        RezLabel = 'Initial Altitude'
        call AddResource( %val(STRHndl), %val('STR '),
     &                    %val(int2(rOldAltitude)), %ref(RezLabel) )
        RezHndl = STRHndl
        call WriteResource( %val(RezHndl) )

c.......setup the ascent selection resource

        STRHndl  = Ne Handle( %val(int4(255) ) )
        RezLabel = 'Ascent Profile'
        call AddResource( %val(STRHndl), %val('STR '),
     &                    %val(int2(rOldAscent)), %ref(RezLabel) )
        RezHndl = STRHndl
        call WriteRes urce( %val(RezHndl) )

c.......setup the cli ate selection resource

        STRHndl  = NewHandle( %val(int4(255) ) )
```

```
    RezLabel = 'Climate File'
    call AddResource( %val(STRHndl), %val('STR '),
   &             %val(int2(rOldClimate)), %ref(RezLabel) )
    RezHndl = STRHndl
    call WriteResource( %val(RezHndl) )

c.......setup the deg West/deg East radio button resource

    STRHndl  = NewHandle( %val(int4(255) ) )
    RezLabel = 'Degree Radio'
    call AddResource( %val(STRHndl), %val('STR '),
   &             %val(int2(rOldDegRadio)), %ref(RezLabel) )
    RezHndl = STRHndl
    call WriteResource( %val(RezHndl) )

c.......setup the m/km radio button resource
    STRHndl  = NewHandle( %val(int4(255) ) )
    RezLabel = 'Distance Radio'
    call AddResource( %val(STRHndl), %val('STR '),
   &             %val(int2(rOldDistRadio)), %ref(RezLabel) )
    RezHndl = STRHndl
    call WriteResource( %val(RezHndl) )

c.......setup the sec/min/hr radio button resource

    STRHndl  = NewHandle( %val(int4(255) ) )
    RezLabel = 'Time Radio'
    call AddResource( %val(STRHndl), %val('STR '),
   &             %val(int2(rOldTimeRadio)), %ref(RezLabel) )
    RezHndl = STRHndl
    call WriteResource( %val(RezHndl) )

c.......now set the flag because we have a target for saving

    iGotOldFile = .true.

   endif

c.....use watch cursor while writing data and resources

   cursorHndl = GetCursor ( %val(int2(4)) )
   call SetCursor ( %val(cursorHndl.CRHDL^.CRPTR^) )

c.....at this point in the logic, the file's forks should both be open;

c.....for each item, we check to see if the open resource file already
c.....contains that resource; if it's there, we change it; otherwise,
c.....we add a resource to the open resource file.

c.....save the mission label resource

   call GetDItem( %val(GetSelection), %val(rMissionLabel),
   &             %ref(DType), %ref(DItem), %ref(tempRect) )
   call GetIText ( %val(DItem) , %val(ItemText) )
   RezHndl = GetResource( %val('STR '), %val(int2(rOldMissionText)) )
   STRHndl = RezHndl
   STRHndl.shdl^.. str = ItemText
   call ChangedResource( %val(RezHndl) )
   call WriteResource( %val(RezHndl) )

c.....save the latitude resource

   call GetDItem( %val(GetSelection), %val(rLatitude),
   &             %ref(DType), %ref(DItem), %ref(tempRect) )
```

```
         call GetIText     val(DItem) , %val(ItemText) )
         RezHndl.bhdl =    tResource(%val('STR '), %val(int2(rOldLatitude)))
         STRHndl = RezH
         STRHndl.shdl^..   r^ = ItemText
         call ChangedRe    rce( %val(RezHndl) )
         call WriteReso    e  ( %val(RezHndl) )

c.....save the longi       e resource

         call GetDItem     al(GetSelection), %val(rLongitude),
       &                   ef(DType), %ref(DItem), %ref(tempRect) )
         call GetIText     val(DItem) , %val(ItemText) )
         RezHndl.bhdl =   tResource(%val('STR '), %val(int2(rOldLongitude)))
         STRHndl = RezH
         STRHndl.shdl^.s   r^ = ItemText
         call ChangedRe    rce( %val(RezHndl) )
         call WriteReso    e  ( %val(RezHndl) )

c.....save the durat       resource

         call GetDItem(    al(GetSelection), %val(rDuration),
       &                   ef(DType), %ref(DItem), %ref(tempRect) )
         call GetIText     val(DItem) , %val(ItemText) )
         RezHndl.bhdl =    tResource(%val('STR '), %val(int2(rOldDuration)))
         STRHndl = RezHn
         STRHndl.shdl^.s   r^ = ItemText
         call ChangedRes   rce( %val(RezHndl) )
         call WriteResou   e  ( %val(RezHndl) )

c.....save the altitu      resource

         call GetDItem(    al(GetSelection), %val(rAltitude),
       &                   ef(DType), %ref(DItem), %ref(tempRect) )
         call GetIText     val(DItem) , %val(ItemText) )
         RezHndl.bhdl =    tResource(%val('STR '), %val(int2(rOldAltitude)))
         STRHndl = RezHn
         STRHndl.shdl^.s   r^ = ItemText
         call ChangedRes   rce( %val(RezHndl) )
         call WriteResou   e  ( %val(RezHndl) )

c.....save the ascent      election resource

         ItemText     =    centSelection
         RezHndl.bhdl =    tResource(%val('STR '), %val(int2(rOldAscent)))
         STRHndl = RezH
         STRHndl.shdl^.    r^ = ItemText
         call ChangedRes   rce( %val(RezHndl) )
         call WriteReso    e  ( %val(RezHndl) )

c.....save the climat      selection resource

         write(CharStri    *) ClimateSelection
         ItemText = Char   ring
         RezHndl.bhdl =    tResource( %val('STR '), %val(int2(rOldClimate)))
         STRHndl = RezH
         STRHndl.shdl          = ItemText
         call ChangedRe    rce( %val(RezHndl) )
         call WriteRes        ( %val(RezHndl) )

c.....save the state       the deg West deg East radio button

         write(CharStri    *  rDegreeSelection
         ItemText = Cha    ring
         RezHndl.bhdl =    Resource( %val('STR '), %val(int2(rOldDegRadio)))
```

```
      STRHndl = RezHndl
      STRHndl.shdl^.sptr^ = ItemText
      call ChangedResource( %val(RezHndl) )
      call WriteResource  ( %val(RezHndl) )

c.....save the state of the m/km radio button

      write(CharString,*) rDistanceSelection
      ItemText = CharString
      RezHndl.bhdl = GetResource( %val('STR '), %val(int2(rOldDistRadio)))
      STRHndl = RezHndl
      STRHndl.shdl^.sptr^ = ItemText
      call ChangedResource( %val(RezHndl) )
      call WriteResource  ( %val(RezHndl) )

c.....save the state of the sec/min/hr radio button

      write(CharString,*) rTimeSelection
      ItemText = CharString
      RezHndl.bhdl = GetResource( %val('STR '), %val(int2(rOldTimeRadio)))
      STRHndl = RezHndl
      STRHndl.shdl^.sptr^ = ItemText
      call ChangedResource( %val(RezHndl) )
      call WriteResource  ( %val(RezHndl) )

c.....save the data fork

      do i = 1, ntrpts
        write(10,*) TofTab(i), LngTab(i), LatTab(i), AltTab(i), JmpTab(i)
      end do

c.....close the resource file

      call CloseResFile( %val(RefNum) )

c.....close the data fork

      close(10)

c.....save the Plot2D-format data file for external use

      open( unit=12, file='BDPS Plot Data', status='new', form='unformatted',
     &      creator='MDoF', filetype='BINA', recordtype='stream' )

c                 123456789-123456
      Name(1) = 'Time            '
      Name(2) = 'Latitude        '
      Name(3) = 'Longitude       '
      Name(4) = 'Altitude        '
      Name(5) = 'Ground Range    '
      Name(6) = 'Wind Azimuth    '
      Name(7) = 'Wind Velocity   '

      write(12) int4(7), int4(-1), int4(1)
      write(12) ( Name(i), i=1,7 )

      do i=1,No_of_Pts
        write(12)     Time_Array( i ),
     &                LAT_ARRAY ( i ),
     &                LON_ARRAY ( i ),
     &                ALT_ARRAY ( i ),
     &                GRANGE_ARRAY ( i ),
     &                WINDAZ_ARRAY ( i ),
     &                WIND_VEL_ARRAY ( i )
```

```
      enddo

      close(12)

c.....reset cursor to arrow

      call SetCursor ( %val(QDG^.Arrow) )

      return
      end


c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

!!S SaveTheMap
c---------------------------------------------------------------------------------
      subroutine SaveTheMap
c---------------------------------------------------------------------------------
c     save the contents of the graphics window into a Pict file

c.....pointer for QuickDraw globals

      common  / QDGPtr /            QDG
      pointer / QDGlobals /         QDG

c.....cursor handle

      record / CursHandle /         CursorHndl

c.....Picture record handle and pointer

      common / pict   /             PictHndl
      record / PicHandle /          PictHndl
      record / PicPtr /             PictPtr

c.....reply record

      record /SFReply/              SFR

c.....prompt location

      record /Point/                PrmptPt

c.....various character strings

      string*255                    Prompt
      string*255                    pfName

c.....Define a FORTRAN style parameter corresponding to the Pascal
c     defined constant nil

      integer*4                     nil

c.....declare integer*2 for Operating System error

      integer*2                     iOSErr

c.....declare integer array used as header for Pict file
```

```

      integer*2                   PictHeader(256)

c.....do loop indices

      integer*4                   i4min
      integer*4                   i4max
      integer*4                   i4

c.....declare temporary integer storage

      integer*4                   ndex4
      integer*4                   noff4
      integer*4                   itemp4
      integer*2                   itemp2
      integer*1                   itemp1

c.....set the Pict file header array

      data PictHeader /
     .  $5049, $4354, $0000, $0000, $0000, $0000, $0000, $0000,
     1  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     2  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     3  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     4  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     5  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     6  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     7  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     8  $0000, $0000, $0000, $0000, $0048, $0000, $0048, $0000,
     9  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     .  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     1  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     2  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     3  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     4  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     5  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     6  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     7  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     8  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     9  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     .  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     1  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     2  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     3  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     4  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     5  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     6  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     7  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     8  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     9  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     .  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000,
     1  $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000/

c.....determine name of Pict file via SFPutFile

      PrmptPt.H = 82
      PrmptPt.V = 64
      Prompt    = 'Save current Map as Pict file:'
      pfName    = 'Map.pict'
      call SFPutFile ( %val(PrmptPt) , %val(Prompt) , %val(pfName) ,
     .                  %val(nil) , %ref(SFR) )
      pfName = SFR.fName

c.....open Pict file
```

```

      if ( SFR.good ) then
         iOSErr = SetVol ( %val(nil) , %val(SFR.vRefNum) )
         open ( unit=10 , file=pfName , creator='MDPL' , filetype='PICT' ,
     .          status='new' , access='sequential' , recordType='stream' ,
     .          form='unformatted' )

c........use watch cursor while saving data

         cursorHndl = GetCursor ( %val(4) )
         call SetCursor ( %val(cursorHndl.CRHDL^.CRPTR^) )

c........pad the Pict file with a header of 256 words

         write(10) PictHeader

c........get pointer to Picture record

         itemp4  = PictHndl
         Pictptr = long ( itemp4 )

c........write number of bytes contained in Picture record

         itemp4  = PictPtr
         itemp2  = word ( itemp4 )
         write(10) itemp2

c........initialize Picture byte counter

         ndex4 = itemp2
         if ( ndex4.lt.2 ) then
            ndex4 = ndex4 + 65536
         end if
         ndex4 = ndex4 - 1

c........write bytes of Picture to Pict file byte by byte

         itemp4 = PictPtr
         do i4 = 2 , ndex4
            itemp1 = byte ( i4 + itemp4 )
            write(10) itemp1
         end do

c........continue here if Picture contains too many bytes

         noff4  = 0
         do while ( itemp1.ne.-1 )
            i4min  = ndex4 + noff4 + 1
            i4max  = ndex4 + noff4 + 65536
            do i4 = i4min , i4max
               itemp1 = byte ( i4 + itemp4 )
               write(10) itemp1
            end do
            noff4  = noff4 + 1
         end do

c........set cursor back to arrow before returning

         call SetCursor ( %val(QDG^.Arrow) )

c........close Pict file

         close ( unit=10 )

      end if

c.....kill the pictu

      call KillPictu   ( %val(PictHndl) )

      return
      end


c.....Load a file of  RUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolB  traps

!!M Inlines.f

c-------------------- ----------------------------------------------------
      subroutine Set  apMenu
c-------------------- ----------------------------------------------------

c.....Options menu f  e

      include 'MapMe   inc'

c.....local string    ables

      string*255            itemName(nMapItems)
      string*255            MenuName

c.....set local stri   variables

      ItemName(itemG  ewDataSet)   = 'Get new data set/G'
      ItemName(itemR  zeTheMap)    = 'Resize the map'
      ItemName(itemN  ap)          = 'New map/N'
      ItemName(itemSa  Map)        = 'Save map/S'
      ItemName(itemRe  aw)         = 'Redraw map/R'
      ItemName(itemDo  e)          = 'Quit MapIt/Q'
      MenuName                     = 'Map'

c.....get handle for  Map' options menu

      if ( iGotMapMen Hndl.eq.0 ) then
         MapMenuHndl     = NewMenu ( %val(MapMenuID) , %val(MenuName) )
         iGotMapMenuH  dl = 1
      end if

c.....append the men  items

      do i = 1 , nMa  tems
         call Append   u ( %val(MapMenuHndl) , %val(ItemName(i)) )
      end do

c.....disable the me    tems not initially available

      call DisableIt      %val(MapMenuHndl) , %val(itemSaveMap)       )
      call DisableI    ( %val(MapMenuHndl) , %val(itemRedraw)         )

c.....draw blank me     ar

      call ClearMenu
      call DrawMenuB
```

```
c.....clear event buffer

      call FlushEvents ( %val(everyEvent) , %val(0) )

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main
c-----------------------------------------------------------------------------
Segment Main
      subroutine SetupTheItem ( theDialog, ItemID, SizeIt, ShowIt, EnableIt,
     .                          SetTheMax, thePosition, ExtraData, StringID )
c-----------------------------------------------------------------------------

!!SETC USINGINCLUDES = FALSE
      implicit none

c.....dialog pointer

      record / DialogPtr / theDialog

c.....item ID and string ID

      integer*2 ItemID
      integer*2 StringID

c.....Boolean arguments

      logical*1 SizeIt
      logical*1 ShowIt
      logical*1 EnableIt
      logical*1 SetTheMax

c.....item location and size

      record / Rect / thePosition

c.....extra data

      integer*4 ExtraData

c.....working values

      record / Rect             tempRect
      record / Handle           DItem
      record / ControlHandle    CItem

      integer*2  DType

      string*255 sTemp

c-----------------------------------------------------------------------------

```

```
c.....get the item handle and size

      call GetDItem( %val(theDialog), %val(ItemID),
     &              %ref(DType), %ref(DItem), %ref(tempRect) )
      CItem.CtlH = DItem.bhdl

c.....check to resize all CDEF connected controls

      if (SizeIt) then
        call SizeControl( %val(CItem), %val(tempRect.right  - tempRect.left),
     &                                 %val(tempRect.bottom - tempRect.top) )
      endif

c.....pass back the location and size

      thePosition = tempRect

      if (ExtraData.ne.0) then
c.......ignore ExtraData for now
        continue
      endif

c.....see if CDEF needs the title set again

      if(StringID.ne.0) then
        call GetIndString( %val(sTemp), %val(StringID), %val(1) )
        call SetCTitle( %val(CItem), %val(sTemp) )
      endif

c.....see if enable or diable the item

      if(EnableIt) then
        call HiliteControl( %val(CItem), %val(0) )
      else
        call HiliteControl( %val(CItem), %val(255) )
      endif

c.....see if set the max

      if(SetTheMax) then
        call SetCtlMax( %val(CItem), %val(12345) )
      endif

c.....see if show it to activate it

      if(ShowIt) then
        call ShowControl( %val(CItem) )
      endif

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c-----------------------------------------------------------------------
      subroutine SetUpTheMap
c-----------------------------------------------------------------------
```

```
c     Interface with the user to obtain map characteristics.

c.....include common block definition files

      include 'CrvDat.inc'
      include 'DefLim.inc'
      include 'FntCom.inc'
      include 'MapCom.inc'
      include 'MapLim.inc'
      include 'TicDat.inc'
      include 'TrjLim.inc'

c.....item stuff

      record / handle /        ItHndl
      record / rect /          ItRect
      integer*4                ItType
      integer*2                ItNmbr
      string*255               ItText

c....."get Map data" dialog interface records

      common / MapSetUp  /     MapSetUpPtr,     iGotMapSetUp
      record / DialogPtr /     MapSetUpPtr
      integer*2                iGotMapSetUp

c.....character strings

      character*255            ChrDat

c.....dialog interface variables ( note that pointers are i*4 )

      integer*4                infront

c.....dialog interface values

      data     infront / -1 /

c.....set dialog font to Times ( it is the most compact )

      FntNam = 'Times'
      call GetFNum ( %val(FntNam) , FntNum )
      call setDAfont ( %val(FntNum) )

c.....Get map set up dialog

      if ( iGotMapSetUp.eq.0 ) then
         MapSetUpPtr  = GetNewDialog ( %val(134) , %val(nil) , %val(inFront) )
         iGotMapSetUp = 1
      end if
      call SetPort ( %val(MapSetUpPtr) )

c.....bring the dialog window to the front

      call ShowWindow ( %val(MapSetUpPtr) )
      call SelectWindow ( %val(MapSetUpPtr) )

c.....Highlight the [illegible] button

      ItNmbr = 1
      call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                     %ref(ItHndl) , %ref(ItRect) )
      call PenSize ( %val(3) , %val(3) )
      call InsetRect ( %ref(ItRect) , %val(-4) , %val(-4) )
      call FrameRoundRect ( %ref(ItRect) , %val(18) , %val(18) )

c.....frame the RESET MAP LIMITS button along with the FULL GLOBAL MAP and
c     SCALE TO DATA radio controls

      call PenSize ( %val(1) , %val(1) )
      call GetDItem ( %val(MapSetUpPtr) , %val(21) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      myLeft = ItRect.left  - 11
      myRite = ItRect.right + 10
      myTop  = ( ItRect.top + ItRect.bottom )/2
      call GetDItem ( %val(MapSetUpPtr) , %val(23) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      myBot  = ItRect.bottom + 10
      call MoveTo ( %val(myLeft+8) , %val(myTop) )
      call LineTo ( %val(myLeft  ) , %val(myTop) )
      call LineTo ( %val(myLeft  ) , %val(myBot) )
      call LineTo ( %val(myRite  ) , %val(myBot) )
      call LineTo ( %val(myRite  ) , %val(myTop) )
      call LineTo ( %val(myRite-8) , %val(myTop) )

c.....display the text items containing map limits

      call DisplayMapLimits

c.....set the 'draw grid lines' control

      ItNmbr = 3
      call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call SetCtlValue ( %val(ItHndl) , %val(GridLines) )

c.....set the 'draw time tics' control

      ItNmbr = 4
      call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call SetCtlValue ( %val(ItHndl) , %val(TimeTics) )

c.....set the 'full global map' radio control

      ItNmbr = 22
      call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call SetCtlValue ( %val(ItHndl) , %val(1-LimitType) )

c.....set the 'scale to data' radio control

      ItNmbr = 23
      call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call SetCtlValue ( %val(ItHndl) , %val(LimitType) )

c.....loop until either the PLOT button or the RETURN button is clicked.
c     Monitor all other relevant events and update the dialog as necessary.

      ItNmbr = 0
      do while ( ItNmbr.ne.1 .and. ItNmbr.ne.2 )

c.........get number of item hit

         call ModalDialog ( %val(nil) , ItNmbr )

c.........alter the 'draw grid lines' user item
```

```
         if ( ItNmbr.eq.3 ) then
            call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                      %ref(ItHndl) , %ref(ItRect) )
            GridLines = 1 - GetCtlValue ( %val(ItHndl) )
            call SetCtlValue ( %val(ItHndl) , %val(GridLines) )
         end if

c........alter the 'draw time tics' user item

         if ( ItNmbr.eq.4 ) then
            call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                      %ref(ItHndl) , %ref(ItRect) )
            TimeTics = 1 - GetCtlValue ( %val(ItHndl) )
            call SetCtlValue ( %val(ItHndl) , %val(TimeTics) )
            if ( TimeTics.eq.1 ) then
               write(ChrDat,*) tDivMj
            else
               ChrDat = ' '
            end if
            ItNmbr = 13
            ItText = ChrDat
            call GetDItem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                      %ref(ItHndl) , %ref(ItRect) )
            call SetIText ( %val(ItHndl) , %val(ItText) )
         end if

c........toggle the 'full global map' and 'scale to data' radio controls

         if ( ItNmbr.eq.22 .or. ItNmbr.eq.23 ) then
            LimitType = 1 - LimitType
            call GetDItem ( %val(MapSetUpPtr) , %val(22) , %ref(ItType) ,
     .                      %ref(ItHndl) , %ref(ItRect) )
            call SetCtlValue ( %val(ItHndl) , %val(1-LimitType) )
            call GetDItem ( %val(MapSetUpPtr) , %val(23) , %ref(ItType) ,
     .                      %ref(ItHndl) , %ref(ItRect) )
            call SetCtlValue ( %val(ItHndl) , %val(LimitType) )
         end if

c........reset the map limits if the 'reset map limits' button is selected

         if ( ItNmbr.eq.21 ) then

c...........use global map limits

            if ( LimitType.eq.0 ) then
               xMapMn = LngMin
               xMapMx = LngMax
               xDivMj = LngDivMj
               xDivMi = LngDivMi
               yMapMn = LatMin
               yMapMx = LatMax
               yDivMj = LatDivMj
               yDivMi = LatDivMi

c..............scale to fit trajectory data

            else
               call AutoScale ( MinLng , MaxLng , ndivmj , xMapMn , xMapMx ,
     .                          xDivMj , xDivMi )
               call AutoScale ( MinLat , MaxLat , ndivmj , yMapMn , yMapMx ,
     .                          yDivMj , yDivMi )
            end if
            call DisplayMapLimits
         end if

      end do

c.....return to main program if RETURN was clicked

      if ( ItNmbr.eq.2 ) then
         call DisposDialog ( %val(MapSetUpPtr) )
         call exit
      end if

c.....get minimum latitude value

      ItNmbr = 5
      call GetDitem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call GetIText ( %val(ItHndl) , %val(ItText) )
      ChrDat = ItText
      if ( ChrDat.ne.' ' ) then
         read(ChrDat,*,iostat=ioflag) tmp1
         if ( ioflag.ne.0 ) then
            tmp1   = 0.0
         end if
      else
         tmp1   = 0.0
      end if
      yMapMn = tmp1

c.....get maximum latitude value

      ItNmbr = 6
      call GetDitem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call GetIText ( %val(ItHndl) , %val(ItText) )
      ChrDat = ItText
      if ( ChrDat.ne.' ' ) then
         read(ChrDat,*,iostat=ioflag) tmp1
         if ( ioflag.ne.0 ) then
            tmp1   = 0.0
         end if
      else
         tmp1   = 0.0
      end if
      yMapMx = tmp1

c.....get latitude major division size

      ItNmbr = 7
      call GetDitem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call GetIText ( %val(ItHndl) , %val(ItText) )
      ChrDat = ItText
      if ( ChrDat.ne.' ' ) then
         read(ChrDat,*,iostat=ioflag) tmp1
         if ( ioflag.ne.0 ) then
            tmp1   = 0.0
         end if
      else
         tmp1   = 0.0
      end if
      yDivMj = tmp1

c.....get latitude minor division size

```

```
      ItNmbr = 8
      call GetDitem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call GetIText ( %val(ItHndl) , %val(ItText) )
      ChrDat = ItText
      if ( ChrDat.ne.' ' ) then
         read(ChrDat,*,iostat=ioflag) tmpl
         if ( ioflag.ne.0 ) then
            tmpl   = 0.0
         end if
      else
         tmpl   = 0.0
      end if
      yDivMi = tmpl

c.....get minimum longitude value

      ItNmbr = 9
      call GetDitem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call GetIText ( %val(ItHndl) , %val(ItText) )
      ChrDat = ItText
      if ( ChrDat.ne.' ' ) then
         read(ChrDat,*,iostat=ioflag) tmpl
         if ( ioflag.ne.0 ) then
            tmpl   = 0.0
         end if
      else
         tmpl   = 0.0
      end if
      xMapMn = tmpl

c.....get maximum longitude value

      ItNmbr = 10
      call GetDitem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call GetIText ( %val(ItHndl) , %val(ItText) )
      ChrDat = ItText
      if ( ChrDat.ne.' ' ) then
         read(ChrDat,*,iostat=ioflag) tmpl
         if ( ioflag.ne.0 ) then
            tmpl   = 0.0
         end if
      else
         tmpl   = 0.0
      end if
      xMapMx = tmpl

c.....get longitude major division size

      ItNmbr = 11
      call GetDitem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call GetIText ( %val(ItHndl) , %val(ItText) )
      ChrDat = ItText
      if ( ChrDat.ne.' ' ) then
         read(ChrDat,*,iostat=ioflag) tmpl
         if ( ioflag.ne.0 ) then
            tmpl   = 0.0
         end if
      else
         tmpl   = 0.0
      end if
      xDivMj = tmpl

c.....get longitude minor division size

      ItNmbr = 12
      call GetDitem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call GetIText ( %val(ItHndl) , %val(ItText) )
      ChrDat = ItText
      if ( ChrDat.ne. ' ' ) then
         read(ChrDat,*,iostat=ioflag) tmpl
         if ( ioflag.ne.0 ) then
            tmpl   = 0.0
         end if
      else
         tmpl   = 0.0
      end if
      xDivMi = tmpl

c.....get time tic increment

      ItNmbr = 13
      call GetDitem ( %val(MapSetUpPtr) , %val(ItNmbr) , %ref(ItType) ,
     .                %ref(ItHndl) , %ref(ItRect) )
      call GetIText ( %val(ItHndl) , %val(ItText) )
      ChrDat = ItText
      if ( ChrDat.ne. ' ' ) then
         read(ChrDat,*,iostat=ioflag) tmpl
         if ( ioflag.ne.0 ) then
            tmpl   = 0.0
         end if
      else
         tmpl   = 0.0
      end if
      tDivMj = tmpl

c.....hide dialog

      call HideWindow ( %val(MapSetUpPtr) )

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c.....Put the following code in the Main segment

!!S Main
c--------------------  ---------------------------------------------------
Segment Main
      subroutine SFGetAscentFile ( iopen , FilNam )
c--------------------  ----------------------------------------  ------------
c     Select an ascent profile using the unit number (icunit).  A successful
c     open will be signalled by returning (iopen=1).

c.....prompt string

```

```
      string*255              Prompt

c.....reply record

      record /SFReply/        SFR

c.....prompt location

      record /Point/          PrmptPt

c.....file information parameters

      record /SFTypeList/     SFT
      record /FInfo/          fndrInfo
      character*4             FilTyp
      character*4             fMaker
      string*255              FilNam

c.....I/O error flags

      integer*2               ioserr,      iopen

c-------------------------------------------------------------------------------

c.....initialize open status flag to zero

      iopen  = 0

c.....set the prompt

      prompt = 'Select an ascent profile'

c.....set prompt box location

      PrmptPt.H = 82
      PrmptPt.V = 64

c.....display files of type ascp ("Ascent Profile") or text

      SFT.SFT(0).OST = 'ascp'
      SFT.SFT(1).OST = 'TEXT'

c.....call the Toolbox

      call SFGetFile( %val(PrmptPt), %ref(Prompt), %val(nil),
     &                %val(int2(2)), %ref(SFT), %val(nil), %ref(SFR) )

c.....Open the file if user selected open

      if ( SFR.good ) then

        iopen  = 1
        FilNam = SFR.fName
        FilTyp = SFR.fType.OST
        ioserr = GetFInfo ( %val(FilNam) , %val(SFR.vRefNum) , %ref(fndrInfo) )
        fMaker = fndrInfo.fdCreator.OST

      else

          call SysBeep( %val(int2(20)) )
        call ExitToShell

      end if

```

```
        return
        end


c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolBox traps

!!M Inlines.f

c---------------------------------------------------------------------------
      subroutine SFOpenFile ( iounit , iopen , FilTyp , fMaker )
c---------------------------------------------------------------------------
c     Open an ASCII file using the unit number [iounit].  A successful open will
c     be signalled by returning [iopen=1].

c.....prompt string

      string*255             Prompt

c.....reply record

      record /SFReply/       SFR

c.....prompt location

      record /Point/         PrmptPt

c.....file information parameters

      record /SFTypeList/    SFT
      record /FInfo/         fndrInfo
      character*(*)          FilTyp
      character*(*)          fMaker
      string*255             FilNam

c.....I/O error flags

      integer*2              ioserr,      ios

c---------------------------------------------------------------------------

c.....initialize open status flag to zero

      iopen  = 0

c.....set the prompt

      prompt = 'Select binary data file'

c.....set prompt box location

      PrmptPt.H = 80
      PrmptPt.V = 64

c.....display files of type TEXT or r*df

      SFT.SFT(0).OST = 'TEXT'
      SFT.SFT(1).OST = 'rtdf'

c.....call the ToolBox

```

```
      call SFGetFil   %val(PrmptPt), %ref(Prompt), %val(nil),
     &                %val(2), %ref(SFT), %val(nil), %ref(SFR) )

c.....Open the file   user selected open

      if ( SFR.good   then
         ioserr = Se  ol ( %val(nil) , %val(SFR.vRefNum) )
         open ( unit  iunit , file=SFR.fName , status='old' , iostat=ios )
         if ( ios.ne   ) then
            write(*,   'Error opening',SFR.fName,': ios=',ios
         else
            iopen  =
            FilNam =  R.fName
            FilTyp =  FR.fType.OST
            ioserr =  etFInfo ( %val(FilNam) , %val(SFR.vRefNum) , %ref(fndrInfo) )
            fMaker = fndrInfo.fdCreator.OST
         end if
      end if

      return
      end
```

```
c.....Load a file of STRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolB  traps

!!M Inlines.f

c-------------------  ----------------------------------------------------------
      subroutine SF   nMissionFile ( iopen , FilTyp , fMaker )
c------------------ ------------------------------------------------------------
c     Open a missio   ile using the unit number [iounit].  A successful open
c     will be signal  ed by returning [iopen=1].

      include 'FileInfo.inc'

c.....prompt string

      string*255              Prompt

c.....reply record

      record /SFReply         SFR

c.....prompt locatio

      record /Point           PrmptPt

c.....file informati   parameters

      record /SFTyp           SFT
      record /FInf            fndrInfo
      character*4             FilTyp
      character*4             fMaker
      string*255              FilNam

c.....I O error fla

      integer*2               iosErr,       ios

c-------------------  ----------------------------------------------------------
```

```

c.....initialize ope  status flag to zero

      iopen  = 0

c.....set the prompt

      prompt = 'Select mission data file'

c.....set prompt box location

      PrmptPt.H = 82
      PrmptPt.V = 64

c.....display files of type 'rtdf'

      SFT.SFT(0).OST = 'rtdf'

c.....call the Toolbo

      call SFGetFile(  val(PrmptPt), %ref(Prompt), %val(int4(nil)),
     &                 val(int2(2)), %ref(SFT), %val(int4(nil)), %ref(SFR) )

c.....Open the file i  user selected open

      if ( SFR.good    hen

c.......open the res  ce fork

        RefNum = Ope  esFile( %ref(SFR.fname) )

        select case   sError)

          case(noErr
            iopen  =
            FilNam =  R.fName
            FilTyp =  R.fType.OST
            ioserr =  etFInfo ( %val(FilNam) , %val(SFR.vRefNum) , %ref(fndrInfo) )
            fMaker =  drInfo.fdCreator.OST

          case defaul
            call SysB ep( %val(int2(20)) )
            call Exit oShell

        end select

      end if

      return
      end
```

```
!!s Sundry
c-----------------------------------------------------------------------
      block data Sundry
c-----------------------------------------------------------------------
c     initialize various parameters via block data

c.....common block files

      include 'CrvDat.inc'
      include 'DefLim.inc'
      include 'MapCom.inc'
      include 'MapLim.inc'
      include 'OptFlg.inc'
      include 'PicGrp.inc'
      include 'TicDat.inc'

c.....various Map curve settings

      data     ipvari      /     1,      0,      0,      0/
      data     ipvard      /     2,      0,      0,      0/
      data     ipvarg      /     0,      0,      0,      0/
      data     idrlin      /     1,      1,      1,      1/
      data     lintyp      /     1,      2,      3,      4/
      data     DshMsk      / $FFFF,  $FFF0,  $FCFC,  $E0E0/
      data     idrsym      /     0,      0,      0,      0/
      data     symtyp      /     1,      2,      3,      4/
      data     ipstep      /     0,      0,      0,      0/
      data     ighoff      /     0,      0,      0,      0/

c.....various Map control settings

      data     GridLines   /     0/
      data     TimeTics    /     0/
      data     LimitType   /     0/

c.....user option flags

      data     oCycle      /     0/
      data     oSave       /     1/
      data     oRedraw     /     2/
      data     oNew        /     3/
      data     oQuit       /     4/

c.....tic mark settings

      data     ndivmj      /    10/
      data     lticmj      /    10/
      data     lticmi      /     5/

c.....picture grouping commands

      data      picGroupBeg /   140/
      data      picGroupEnd /   141/

c.....global Map limits

      data      LngMin       / -180.0 /
      data      LngMax       /  180.0 /
      data      LngDivMj     /   30.0 /
      data      LngDivMi     /   10.0 /

      data      LatMin       /  -90.0 /
      data      LatMax       /   90.0 /
      data      LatDivMj     /   30.0 /
      data      LatDivMi     /   10.0 /

c.....default Map limits

      data      xMapMn       / -180.0 /
      data      xMapMx       /  180.0 /
      data      xDivMj       /   30.0 /
      data      xDivMi       /   10.0 /

      data      yMapMn       /  -90.0 /
      data      yMapMx       /   90.0 /
      data      yDivMj       /   30.0 /
      data        yDi         /   10.0 /

      data        tMa           /    0.0 /
      data        tMa           / 1000.0 /
      data        tD            /  100.0 /

      end
```

```
c.....Load a file of  IRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc

c.....Load the ToolB   traps

!!M Inlines.f

c.....Put the follow  g code in the Main segment

!!S Main

c------------------ --------------------------------------------------------------
Segment Main
      logical functi   TrapAvailable( tNumber, tType)
c------------------ --------------------------------------------------------------
c     This is my imp   entation of the TrapAvailable function
c     from the DTS S  ple application.

      implicit none
      integer*2 tNum          ! trap number, defined in traps.f
      integer*1 tTyp       ! trap type (enumerated in c/pascal)
      integer*1 Tool   ap  ! trap type for comparison
      data ToolTrap    / ! second enumerated type should be "1"?
      logical check   heck2

c.....common block d   inition files

      include 'Globa  .inc'

c------------------ --------------------------------------------------------------

      if ( (tType.eq  oolTrap) .and.
     1    ((gMac.mac  neType).gt.envMachUnknown) .and.
     2    ((gMac.mac  neType).lt.envMacII)) then
              tNumbe  = iand(tNumber,$03FF)
              if(tNu  er.gt.$01FF) tNumber=tUnimplemented
      end if

      check1 = NGetI   Address(tNumber, tType)
      check2 = GetI   ddress(tUnimplemented)
      if(check1.eq..   k2)then
          TrapAvail   e = .false.
      else
          TrapAvail   e = .true.
      endif

      return
      end
```

```
c.....Load a file of  TRUCTURE and PARAMETER definitions at compile time

!!G toolbox2.finc
```

```

c.....Load the ToolBox traps

!!M Inlines.f

c-----------------------------------------------------------------------------
          subroutine TrashBitMap ( oldOffScreen , myBitH )
c-----------------------------------------------------------------------------
c         get rid of everything associated with the off screen bit map

          record / GrafPtr /           oldOffScreen
          record / Handle /            myBitH

c.....close the bit map port and dispose of the associated pointers

          call ClosePort     ( %val(oldOffScreen) )
          call DisposPtr     ( %val(oldOffScreen.GrafP^.portBits.baseAddr) )
          call DisposPtr     ( %val(oldOffScreen) )
          call HUnlock       ( %val(myBitH) )
          call DisposHandle  ( %val(myBitH) )

          return
          end
```

10.4 BDPS REZ SOURCE CODE

This section contains the BDPS file for input to the MPW resource compiler ("Rez"). Most of the information in this file is directly understandable upon inspection. However, for the sake of completeness, the file also contains machine-readable representations of several icons, a "PICT" file, and a popup-menu handler, all of which are used in building and operating the BDPS program.

```
/*  bdps.r
      Rez file for balloon drift pattern simulation */

#include "systypes.r"
#include "types.r"

#define rMenuBar               128     /* application's menu bar */
#define mApple                 128     /* Apple menu */
#define mFile                  129     /* File menu */
#define mEdit                  130     /* Edit menu */
#define mMap                   131     /* Map menu */

#define rAboutAlert            128     /* "About BDPS" alert box */
#define rRunStatus             129     /* Run status dialog */
#define rMapSetup              134     /* Map setup dialog */
#define rResizeMap             135     /* Resize Map dialog */
#define rRunSetupDLOG          256     /* Run Setup dialog */
#define rBalloonPICT           1001    /* PICT for "About BDPS" */

/* these #defines are used to set enable/disable flags of a menu */

#define AllItems               0b1111111111111111111111111111111    /* 31 flags */
#define NoItems                0b0000000000000000000000000000000
#define MenuItem1              0b0000000000000000000000000000001
#define MenuItem2              0b0000000000000000000000000000010
#define MenuItem3              0b0000000000000000000000000000100
#define MenuItem4              0b0000000000000000000000000001000
#define MenuItem5              0b0000000000000000000000000010000
```

```
#define MenuItem6           0b0000000000000000000000000100000
#define MenuItem7           0b0000000000000000000000001000000
#define MenuItem8           0b0000000000000000000000010000000
#define MenuItem9           0b0000000000000000000000100000000
#define MenuItem10          0b0000000000000000000001000000000
#define MenuItem11          0b0000000000000000000010000000000
#define MenuItem12          0b0000000000000000000100000000000
#define         MenuItem13  0b0000000000000000001000000000000   /* 13 */
#define         MenuItem14  0b0000000000000000010000000000000   /* 14 */
#define         MenuItem15  0b0000000000000000100000000000000   /* 15 */
#define         MenuItem16  0b0000000000000001000000000000000   /* 16 */
#define         MenuItem17  0b0000000000000010000000000000000   /* 17 */
#define         MenuItem18  0b0000000000000100000000000000000   /* 18 */
#define         MenuItem19  0b0000000000001000000000000000000   /* 19 */
#define         MenuItem20  0b0000000000010000000000000000000   /* 20 */
#define         MenuItem21  0b0000000000100000000000000000000   /* 21 */
#define         MenuItem22  0b0000000001000000000000000000000   /* 22 */
#define         MenuItem23  0b0000000010000000000000000000000   /* 23 */
#define         MenuItem24  0b0000000100000000000000000000000   /* 24 */
#define         MenuItem25  0b0000001000000000000000000000000   /* 25 */
#define         MenuItem26  0b0000010000000000000000000000000   /* 26 */
#define         MenuItem27  0b0000100000000000000000000000000   /* 27 */
#define         MenuItem28  0b0001000000000000000000000000000   /* 28 */
#define         MenuItem29  0b0010000000000000000000000000000   /* 29 */
#define         MenuItem30  0b0100000000000000000000000000000   /* 30 */
#define         MenuItem31  0b1000000000000000000000000000000   /* 31 */

type 'MDoF' as 'STR ';                  /*  crcH is the signature */
resource 'MDoF' (0)  {                  /* the creator resource ID must be zero */
        "Balloon Drift Pattern Simulation 1.0 copyright 1991"
};

/* use an MBAR resource to conveniently load all menus */

resource 'MBAR' (rMenuBar, "Balloon Drift menu bar", preload) {
        { mApple, mFile, mEdit, mMap};                 /* four menus */
};
resource 'MENU' (mApple, "Apple menu", preload) {
        mApple, textMenuProc,
        AllItems & ~MenuItem2, /* disable dashed line, enable About and DAs */
        enabled, apple,
        {
                "About BDPS...",
                        noicon, nokey, nomark, plain;
                "-",
                        noicon, nokey, nomark, plain
        }
};

resource 'MENU' (mFile, "File menu", preload) {
        mFIle, textMenuProc
        MenuItem12,                                    /* enable Quit only, program enables
others */
        enabled, "File",
        {
                "New Mission
                        noicon, "[illegible]", nomark, plain;
                "Open Mission
                        noicon, "[illegible]", nomark, plain;
                "-",
                        noicon, nokey, nomark, plain;
                "Close",
                        noicon, "[illegible]", nomark, plain;
                "Save",
                        noicon   S", nomark, plain;
                "Save As…",
                        noicon   okey, nomark, plain;
                "Revert",
                        noicon   okey, nomark, plain;
                "-",
                        noicon   okey, nomark, plain;
                "Page Setup…"
                        noicon   okey, nomark, plain;
                "Print…",
                        noicon   okey, nomark, plain;
                "-",
                        noicon   okey, nomark, plain;
                "Quit",
                        noicon   Q", nomark, plain
        }
};

resource 'MENU' (mEdit, "Ed   enu", preload) {
        mEdit, textMenuProc,
        NoItems,                  disable everything, program does the enabling */
        enabled, "Edit",
         {
                "Undo",
                        noicon    ", nomark, plain;
                "-",
                        noicon    key, nomark, plain;
                "Cut",
                        noicon   X", nomark, plain;
                "Copy",
                        noicon   C", nomark, plain;
                "Paste",
                        noicon   V", nomark, plain;
                "Clear",
                        noicon   okey, nomark, plain
        }
};

resource 'MENU' (mMap, "Map   nu", preload) {
        mMap, textMenuProc,
        NoItems,                  * disable everything, program does the enabling */
        enabled, "Map",
         {
                "Get New Data   t",
                        noicon   okey, nomark, plain;
                "Resize the M   ,
                        noicon   okey, nomark, plain;
                "New Map",
                        noicon   okey, nomark, plain;
                "Save Map",
                        noicon   okey, nomark, plain;
                "Redraw",
                        noicon   okey, nomark, plain;
                "Done",
                        noicon   okey, nomark, plain
        }
};

resource 'MENU' (41,"Wind M   :",preload) {
        41, textMenuProc,

        AllItems,                                       /* Enable and disable of items */
        enabled, "Wind Model                            /* Menu list name */
```

```
            {
            /*  1 */  "January Climate    ",
            noIcon, noKey, noMark,  plain,
            /*  2 */  "February Climate    ",
            noIcon, noKey, noMark,  plain,
            /*  3 */  "March Climate    ",
            noIcon, noKey, noMark,  plain,
            /*  4 */  "April Climate    ",
            noIcon, noKey, noMark,  plain,
            /*  5 */  "May Climate    ",
            noIcon, noKey, noMark,  plain,
            /*  6 */  "June Climate    ",
            noIcon, noKey, noMark,  plain,
            /*  7 */  "July Climate    ",
            noIcon, noKey, noMark,  plain,
            /*  8 */  "August Climate    ",
            noIcon, noKey, noMark,  plain,
            /*  9 */  "September Climate    ",
            noIcon, noKey, noMark,  plain,
            /* 10 */  "October Climate    ",
            noIcon, noKey, noMark,  plain,
            /* 11 */  "November Climate    ",
            noIcon, noKey, noMark,  plain,
            /* 12 */  "December Climate    ",
            noIcon, noKey, noMark,  plain,
            /* 13 */  "Forecast    ",
            noIcon, noKey, noMark,  plain,
            }
        };

resource 'MENU' (42,"Popup menu",preload) {
        42, textMenuProc,

        AllItems,                          /* Enable and disable of items */
        enabled, "Popup menu",             /* Menu list name */
```

```
                {

                /*  1 */  "Untitled274     ",

                noIcon, noKey, noMark,  plain,

                }

        };


resource 'MENU' (43,"Input Ascent Profile:",preload) {
        43, textMenuProc,

        AllItems,                                       /* Enable and disable of items */
        enabled, "Input Ascent Profile:",    /* Menu list name */
                {

                /*  1 */  "Ascent Profile     ",

                noIcon, noKey, noMark,  plain,

                /*  2 */  "User-defined file...     ",

                noIcon, noKey, noMark,  plain,

                }

        };


resource 'BNDL' (128) {
        'MDoF',                 /* the signature of this application */
        0,                      /* the creator resource ID must be 0 */
        {       'ICN#',
                {       0, 128
                        1, 129
                        2, 130
                        3, 131
                },
                'FREF',
                {       0, 128
                        1, 129
                        2, 130
                        3, 131
                }
        }
};

resource 'ICN#' (128, "BDPS Application") {
        {       /* array: 2 elements */
                /* [1] */
                $"0000 0000 [illegible] 0000 0000 0000 0003 F000"
                $"00BD EC00 [illegible] 3200 0016 3900 0566 3900"
                $"002C 3C80 [illegible] 3C80 02EC 3C80 002C 3C80"
                $"0024 3C80 [illegible] 3D00 0016 3900 000A 3A00"
                $"00AF 3400 [illegible] 3800 0002 0800 002E 0800"
                $"0001 F000 [illegible] 1000 0017 1000 0001 F000"
                $"0001 F000 [illegible] F000 0001 F000 0000 E0",
                /* [2] */
                $"3FFF FFFC [illegible] FFFE FFFF FFFF FFFF FFFF"
                $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
                $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
                $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
                $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
                $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
                $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
                $"FFFF FFFF FFFF FFFF 7FFF FFFE 3FFF FFFC"
        }
};

resource 'ICN#' (129, "runtime data file") {
        {       /* array: 2 elements */
                /* [1] */
                $"FFFF FF00 8000 0180 B81F 8140 A42F 6120"
                $"B859 9110 A451 C908 B931 CDFC 8161 E4FE"
                $"B961 E406 A561 E406 A561 E406 A521 E006"
                $"B8A1 E806 80F1 C806 B851 D006 A439 A006"
                $"B817 4006 A010 4006 A00F 8006 8008 8006"
                $"9C08 8006 A00F 8006 BC0F 8006 840F 8006"
                $"B807 0006 8000 0006 A299 A696 B6B3 2AD6"
                $"AA99 AAB6 A2B0 2C96 FFFF FFFE 7FFF FFFE",
                /* [2] */
                $"FFFF FF00 FFFF FF80 FFFF FFC0 FFFF FFE0"
                $"FFFF FFF0 FFFF FFF8 FFFF FFFC FFFF FFFE"
                $"FFFF FFFE FFFF FFFE FFFF FFFE FFFF FFFE"
                $"FFFF FFFE FFFF FFFE FFFF FFFE FFFF FFFE"
                $"FFFF FFFE FFFF FFFE FFFF FFFE FFFF FFFE"
                $"FFFF FFFE FFFF FFFE FFFF FFFE FFFF FFFE"
                $"FFFF FFFE FFFF FFFE FFFF FFFE FFFF FFFE"
                $"FFFF FFFE FFFF FFFE FFFF FFFE 7FFF FFFE"
        }
};

resource 'ICN#' (130, "atmosphere data") {
        {       /* array: 2 elements */
                /* [1] */
                $"E031 4198 8DCB 1244 E804 0802 1080 000A"
                $"9080 0011 8880 0811 E2C2 0841 0269 0C21"
                $"4200 B061 41AA 0AB2 40D5 555C 007A EAA0"
                $"A00D B6C4 E007 0F00 A000 1048 A004 0180"
                $"0040 C290 4088 0000 A000 8100 E102 0000"
                $"A000 0840 0241 0200 E040 2080 4010 0400"
                $"4404 2000 4020 0200 0100 8000 E008 0000"
                $"8400 0000 C003 8CE6 8822 5E4F E003 9249",
                /* [2] */
                $"0031 C198 0FFB F3FC 1FFF FFFE 1FFF FFFE"
                $"1FFF FFFF 0FFF FFFF 03FF FFFF 03FF FFFF"
                $"03FF FFFF 01FF FFFE 00FF FFFC 007F FFE0"
                $"000F BFC4 0007 0F00 0000 1048 0004 0180"
                $"0040 C290 0088 0000 0000 8100 0102 0000"
                $"0000 0840 0241 0200 0040 2000 0010 0000"
                $"0404 2880 0020 0000 0100 0000 0008 8200"
                $"0400 0000 0000 0000 0820"
        }
};

resource 'ICN#' (131, "ascent profile") {
        {       /* array: 2 elements */
                /* [1] */
                $"FFFF FFFF 8000 0001 8C66 74B9 9289 4691"
                $"9EE8 6791 9229 4591 92C6 7491 8000 0001"
                $"9FFF FFF9 9000 0009 9000 0009 9040 0009"
                $"90AF FFE9 9100 0009 9100 0009 9300 0009"
                $"9200 0009 9200 0009 9400 0009 9400 0009"
                $"9800 0009 9800 0009 9800 0009 9FFF FFF9"
                $"8000 0001 B9C6 751D A529 4511 B9C9 6519"
                $"A149 4511 A126 45DD 8000 0001 FFFF FFFF",
                /* [2] */
                $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
                $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
                $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
                $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
                $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
                $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
                $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
                $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
        }
};

resource 'FREF' (128) {
        'APPL',
        0,
        ""
};

resource 'FREF' (129) {
        'rtdf',
        1,
        ""
};

resource 'FREF' (130) {
        'gram',
        2,
        ""
};

resource 'FREF' (131) {
        'ascp',
        3,
        ""
};

/* this ALRT and DITL are used as an About screen */

resource 'ALRT' (rAboutAlert, "About Box", purgeable) {
        {40, 20, 338, 506},
        rAboutAlert,
        { /* array: 4 elements */
                /* [1] */
                OK, visible, silent,
                /* [2] */
                OK, visible, silent,
                /* [3] */
                OK, visible, silent,
                /* [4] */
                OK, visible, silent
        }
};

resource 'DITL' (rAboutAlert, "About Box", purgeable) {
        {       /* array DITLarray: 2 elements */
                /* [1] */
                {247, 352, 27[illegible], 452},
                Button {
                        enabled
                        "OK"
                },
                /* [2] */
                {-1, -1, 290, 476},
                Picture {
                        disab
                        rBall    ist
                }
        ;
};

 * This is the definition    the run status dialog */

resource  DLOG' (rRunStatus   Run Status Dialog") {
        {40, 40, 123, 368},             /* Top Left Bottom Right */
        dBoxProc,
        invisible,
        noGoAway,
        0x0,
        rRunStatus,
        ""
};

/* This is the item list for  he run status dialog */

resource 'DITL' (rRunStatus, "Run Status Dialog") {
        {       /* array DITLa ray: 3 elements */
                /* [1] */
                {14, 33, 50,  6},
                StaticText {
                        disable ,
                        "Gener  ing balloon drift pattern data..."
                },
                /* [2] */
                {44, 53, 72, 2 5},
                StaticText {
                        disable ,
                        "Percer  complete:"
                },
                /* [3] */
                {44, 188, 60, 70},
                EditText {
                        enabled,
                        "0.0"
                }
        }
};

/* This is the definition for the run setup dialog,  "Untitled" */

resource 'DLOG' (rRunSetupDLOG, "Run Setup Dialog", purgeable) {  /* Dialog */
        { 30, 10, 332, 501 }            /* Top Left Bottom Right */
        dBoxProc, Invisible, noGoAway,256, rRunSetupDLOG,    /* ProcID, visible, noGoAway,
RefCon, DITL */
        "Untitled"
        };

 * This is the item list for  e run setup dialog*/

resource 'DITL' (rRunSetupDL    "Run Setup Dialog", purgeable)
        {   /* DITL array of    tems */

        /* [1] */
        { 268, 396, 288, 475            /* Top Left Bottom Right */
        Button {
                enabled,
                "Close"
        },
/* [2] */
{ 184, 396, 204, 476            /* Top Left Bottom Right */
Button {
        enabled,
        "Run"
},

/* [3] */
{ 240, 396, 260, 476            /* Top Left Bottom Right */
Button {
        enabled,
        "Map"
},

/* [4] */
{ 212, 396, 232, 476            /* Top Left Bottom Right */
Button {
        enabled,
        "Save"
},

/* [5] */
{ 89, 246, 104, 326 },          /* Top Left Bottom Right */
RadioButton {
        enabled,
        "deg West"
},

/* [6] */
{ 89, 338, 104, 418 },          /* Top Left Bottom Right */
RadioButton {
        enabled,
        "deg East"
},

/* [7] */
{ 117, 246, 132, 282 },         /* Top Left Bottom Right */
RadioButton {
        enabled,
        "m"
},

/* [8] */
{ 117, 294, 132, 338 },         /* Top Left Bottom Right */
RadioButton {
        enabled,
        "km"
},

/* [9] */
{ 145, 246, 160, 294 },         /* Top Left Bottom Right */
RadioButton {
        enabled,
        "sec"
},

/* [10] */
{ 145, 294, 160, 342            /* Top Left Bottom Right */
RadioButton {
        enabled,
        "min"
},

/* [11] */
```

```
{ 145, 342, 160, 378 },          /* Top Left Bottom Right */
RadioButton {
        enabled,
        "hr"
},

/* [12] */
{ 65, 104, 80, 168 },            /* Top Left Bottom Right */
StaticText {
        disabled,
        "Latitude:"
},

/* [13] */
{ 89, 94, 104, 174 },            /* Top Left Bottom Right */
StaticText {
        disabled,
        "Longitude:"
},

/* [14] */
{ 65, 263, 80, 343 },            /* Top Left Bottom Right */
StaticText {
        disabled,
        "deg"
},

/* [15] */
{ 141, 61, 157, 169 },           /* Top Left Bottom Right */
StaticText {
        disabled,
        "Flight Duration:"
},

/* [16] */
{ 69, 18, 101, 78 },             /* Top Left Bottom Right */
StaticText {
        disabled,
        "Launch Position"
},

/* [17] */
{ 117, 62, 133, 170 },           /* Top Left Bottom Right */
StaticText {
        disabled,
        "Initial Altitude:"
},

/* [18] */
{ 19, 16, 50, 68 },              /* Top Left Bottom Right */
StaticText {
        disabled,
        "Mission Label
},

/* [19] */
{ 19, 83, 53, 465 },             /* Top Left Bottom Right */
EditText {
        enabled,
        "Wallops Island Flight - Z configuration (this text may be used to identify the
mission represented by this data)"
},

/* [20] */
```

```
        { 64, 185, 78, 235              /* Top Left Bottom Right */
        EditText {
                enabled,
                "54.45"
        },

        /* [21] */
        { 90, 185, 104, 235             /* Top Left Bottom Right */
        EditText {
                enabled,
                "154.45"
        },

        /* [22] */
        { 116, 185, 130, 235            /* Top Left Bottom Right */
        EditText {
                enabled,
                "154.45"
        },

        /* [23] */
        { 143, 185, 157, 235            /* Top Left Bottom Right */
        EditText {
                enabled,
                "154.45"
        },

        /* [24] */
        { 175, 93, 195, 343             /* Top Left Bottom Right */
        Control {
                enabled,
                133                             /* Popup Resource ID linked to */
        },

        /* [25] */
        { 204, 40, 224, 345             /* Top Left Bottom Right */
        Control {
                enabled,
                133                             /* Popup Resource ID linked to */
        },

        }
};

/* This is the definition f    he map setup dialog */

resource 'DLOG' (rMapSetup,    p Setup Dialog", purgeable) {  /* Dialog */
        { 41, 11, 330, 390 }          /* Top Left Bottom Right */
        documentProc, visibl-  noGoAway,0x0, rMapSetup,   /* ProcID, visible, noGoAway,
RefCon, DITL */
        "Map Set Up"
        };

/* This is the item list fo    e map setup dialog*/

resource 'DITL' (rMapSetup,    p Set Up", purgeable) {
        {       /* array DIT    y: 23 elements */
                /* [1] */
                {11, 11, 35.
                Button {
                        enabl
                        "PLOT"
                },
                /* [2] */
```

```
{46, 11. 7
Button {
        enable
        "RETUR
},
/* [3] */
{11, 101, 31.      ),
CheckBox {
        enable
        "draw     . lines"
},
/* [4] */
{36, 101, 55,      ),
CheckBox {
        enable
        "draw     tics"
},
/* [5] */
{116, 111, 1      10},
EditText {
        disabl
        ""
},
/* [6] */
{151, 111, 1      10},
EditText {
        disabl
        ""
},
/* [7] */
{186, 111, 21.    10},
EditText {
        disabl
        ""
},
/* [8] */
{221, 111, 24     10},
EditText {
        disabl
        ""
},
/* [9] */
{116, 221, 14     20},
EditText {
        disabl
        ""
},
/* [10] */
{151, 221, 17     20},
EditText {
        disable
        ""
},
/* [11] */
{186, 221, 21     20},
EditText {
        disable
        ""
},
/* [12] */
{221, 221, 245.   20},
EditText {
        disablec
        ""
```

```
},
/* [13] */
{256, 111, 280, 210},
EditText {
        disabled,
        ""
},
/* [14] */
{116, 6, 140, 105},
StaticText {
        disabled,
        "minimum axis value"
},
/* [15] */
{151, 6, 175, 105},
StaticText {
        disabled,
        "maximum axis value"
},
/* [16] */
{186, 6, 210, 105},
StaticText {
        disabled,
        "major division size"
},
/* [17] */
{221, 6, 245, 105},
StaticText {
        disabled,
        "minor division size"
},
/* [18] */
{261, 6, 285, 105},
StaticText {
        disabled,
        "time tic increment"
},
/* [19] */
{96, 111, 110, 190},
StaticText {
        disabled,
        "latitude"
},
/* [20] */
{96, 221, 110, 300},
StaticText {
        disabled,
        "longitude"
},
/* [21] */
{11, 241, 35, 360},
Button {
        enabled,
        "RESET MAP LIMITS"
},
/* [22] */
{41, 241, 60, 360},
RadioButton {
        enabled,
        "full global map"
},
/* [23] */
{66, 241, 85, 360},
RadioButton {
```

```
                        enabl
                        "scal     data"
                }
        }
};

/* This is the definition f     e map resize dialog */

resource 'DLOG' (rResizeMap.   esize Map Dialog", purgeable) {   /* Dialog */
        { 41, 11, 170, 370              /* Top Left Bottom Right */
        documentProc, visibl    oGoAway,0x0, rResizeMap,    /* ProcID, visible, noGoAway,
RefCon, DITL */
        "Resize Map"
        };

/* This is the item list fo     e map resize dialog*/

resource 'DITL' (rResizeMap.    esize Map", purgeable) {
        {       /* array DITL     y: 6 elements */
                /* [1] */
                {11, 21, 35.
                Button {
                        enable
                        "OK"
                },
                /* [2] */
                {46, 21, 70,
                Button {
                        enable
                        "RESET
                },
                /* [3] */
                {11, 236, 35,     3},
                EditText {
                        disabl
                        ""
                },
                /* [4] */
                {46, 236, 70,     0},
                EditText {
                        disable
                        ""
                },
                /* [5] */
                {11, 126, 35,     0},
                StaticText {
                        disabl
                        "map w    ow width"
                },
                /* [6] */
                {46, 126, 70,     0},
                StaticText {
                        disabl
                        "map w    w height"
                }
        }
};


resource 'STR#' (281 , "Lat     e_Label") {
{  /*                                           for Static text */
/* [1] */
"Latitude:"
        }
};

resource 'STR#' (282 , "Lon-    .e_Label") {
{     *                                         for Static text */
 * [1] *
"Longitude:"
        }
};

resource 'STR#' (284 , "Lat     .e_Units") {
{   /*                                          for Static text */
/* [1] */
"deg"
        }
};

resource 'STR#' (287 , "Dur.    n_Label") {
{   /*                                          for Static text */
/* [1] */
"Flight Duration:"
        }
};

resource 'STR#' (274 , "Laun    Label2") {
{   /*                                          for Static text */
/* [1] */
"Launch Position"
        }
};

resource 'STR#' (290 , "Alti    .e_Label") {
{   /*                                          for Static text */
/* [1] */
"Initial Altitude:"
        }
};

resource 'STR#' (305 , "Miss.  n_Label") {
{   /*                                          for Static text */
/* [1] */
"Mission Label:"
        }
};

resource 'STR#' (280 , "Latit  e") {
{   /*                                          for default edit text */
/* [1] */
"54.45"
        }
};

resource 'STR#' (283 , "Latit  e2") {
{   /*                                          for default edit text */
/* [1] */
"154.45"
        }
};

resource 'STR#' (270 , "Latit  e3") {
{   /*                                          for default edit text */
/* [1] */
"154.45"
        }
};
```

```
resource 'STR#' (291 , "Lat.   .e4") {
{  *                                    for default edit text */
 * [1] *
"154.45"
        }
};

resource 'STR#' (304 , "Mi     _Text") {
{  /*                                   for default edit text */
/* [1] */
"Wallops Island flight - 2      .iguration (this text may be used to identify the mission
represented by this data)"
        }
};

resource 'PICT' (rBalloonPI    "Erics About BDPS picture") {
        14210,
        {-1, -1, 286, 508},
        $"1101 A000 82A1 00E   04 000C 0080 0100 0AFF FFFF FF01 1E01 FC09 8822 8822 8822"
        $"8822 3100 FF00 00    00 3109 FFFF FFFF FFFF FFFF 38A1 00B6 0004 0001 0001 0700"
        $"0000 0023 0000 A1    00 0400 0C00 8022 0106 0006 0000 A000 A0A1 00A4 0002 070C"
        $"0900 0000 0000 00    71 0092 0102 0004 011A 002C 0104 0007 0105 0007 0107 0006"
        $"0108 0005 0109 00   10C 0004 010E 0004 0110 0004 0112 0005 0113 0005 0115 0006"
        $"0117 0007 0118 00    1A 0009 011A 0018 011A 0028 0118 0029 0117 002A 0115 002B"
        $"0114 002B 0112 00    10 002C 010E 002C 010D 002C 010A 002C 0108 002B 0107 002B"
        $"0105 002A 0104 00    02 0027 0102 0018 0102 0008 0102 0008 0104 0007 0100 0A00"
        $"0000 0000 0000 00    01 0001 09FF FFFF FFFF FFFF FF22 0102 0008 FE04 23FE 0423"
        $"0007 2302 0523 03     0 0023 1F00 2300 0023 02FC 2302 FB23 00FA 23FE FB23 FDFC"
        $"2300 0023 E100 23    23 0000 0A00 0000 0000 0000 0084 000A 0000 0000 0000 0000"
        $"A000 A301 000A FF    FF 011E 01FC 8400 0A00 0000 0000 0000 0022 0104 0007 0001"
        $"23FF 0223 FF01 23    23 FF03 2300 0223 0002 2301 0223 0001 2301 0223 0102 2300"
        $"0123 0202 230F 00    00 2301 FE23 01FF 2301 FE23 00FF 2301 FE23 00FE 2300 FE23"
        $"00FF 2300 FD23 FF   300 FF23 FFFE 23FF FF23 FEFE 23F1 0023 F000 2300 0023 FF02"
        $"A000 A1A1 00B6 00    01 0001 0700 0000 0023 0000 A100 9600 0C01 0000 0002 0000"
        $"0000 0000 00A1 00   008 FFFB 0000 0006 0000 0100 0A01 0600 1201 1700 212C 000B"
        $"0010 0850 616C 61    6E 6F03 0010 0D00 1028 0113 0013 0152 A000 97A1 0096 000C"
        $"0100 0000 0200 00    00 0000 A100 9A00 08FF FB00 0000 0600 0001 000A 0106 0006"
        $"0117 0015 2801 13    01 43A0 0097 A100 9600 0C01 0000 0002 0000 0000 0000 00A1"
        $"009A 0008 FFFB 00   006 0000 0100 0A01 0600 1D01 1700 2C29 1701 43A0 0097 A100"
        $"9600 0C01 0000 00   000 0000 0000 00A1 009A 0008 0003 0000 003B 0000 0100 0A01"
        $"0200 3301 1D00 AC   00C 2801 0C00 3410 436F 6C65 6D61 6E20 5265 7365 6172 6368"
        $"0D00 0D29 6801 0D   00C 2801 1A00 3413 4875 6E74 7376 696C 6C65 2044 6976 6973"
        $"696F 6EA0 0097 A1    00 0400 4000 8001 000A FFFF FFFF 011E 01FC 2200 0301 0E00"
        $"00A0 00A0 A100 A4   201 4001 000A 0000 0000 0000 0000 0700 0100 0122 0009 0116"
        $"F8FA 23E4 FD23 F1    F8 0923 F1FF 23F2 F223 E4FB 23F3 0E23 FF0E 23F8 0423 F901"
        $"23FC 0623 ECFF 23    23 F2FF 23F9 04A0 00A3 0100 0AFF FFFF FF01 1E01 FC22 0009"
        $"0116 FEFF 23FB FD   AFE 23FA FF23 FD00 23FF 0023 FC00 23FC 0023 FE00 23FD 0123"
        $"FF00 23FC 0223 FF   3FF 0123 FE02 23FD 0223 FC03 23FE 0223 FF01 23FF 0123 FE02"
        $"23FD 0123 FD00 23   023 FE00 23FC FF23 FCFE 23FD FE23 FEFE 23FE FE23 FBFD 23FB"
        $"FE23 FBFF 23FC FF   F00 23FB FF23 FD00 23FF 0023 FC01 23FE 0123 FD01 23FF 0123"
        $"FF01 23FF 0123 FD0  3FF 0223 FF04 2300 0223 0001 23FF 0323 FF02 23FE 0323 FF00"
        $"23FE 0123 FC02 23F   023 FF00 23FF 0023 FF01 23FF 0123 FF01 2300 0023 FD02 23FD"
        $"0123 FD00 23FD 002   E00 23FC 0023 FD00 23FD 0123 FF01 23FF 0123 FF00 23FD 0023"
        $"FC00 23FF 0023 FE0   2FC 0123 FD01 23FD 0123 FE01 A000 A1A1 00B6 0004 0001 0001"
        $"0700 0000 0023 000   100 B600 0400 4000 8022 0011 0104 0000 A000 A0A1 00A4 0002"
        $"0340 0988 2288 228   298 2271 0302 000B 00D5 00B4 0120 000B 0114 000C 0112 000D"
        $"010F 000E 010C 001   108 0011 0105 0011 0104 0012 0102 0013 0100 0013 0100 0014"
        $"00FE 0017 00FA 001   0F8 001A 00F6 001E 00F3 0020 00F2 0021 00F2 0022 00F2 0023"
        $"00F2 0024 00F3 002   0F4 0023 00F6 0021 00F9 0020 00FA 0020 00FA 001F 00FB 001F"
        $"00FC 001F 00FD 001F  0FD 001F 00FD 0020 00FD 0022 00FC 0026 00FA 0028 00F8 002B"
        $"00F6 002E 00F4 0033  0F2 0038 00F0 003A 00F0 003C 00F0 0041 00EF 0046 00EF 004B"
        $"00EF 004D 00EF 005   0EF 0055 00F0 0059 00F1 005D 00F2 005F 00F3 0061 00F4 0064"
        $"00F6 0067 00F8 006A  0FB 006B 00FC 006D 00FF 0071 0104 0072 0107 0073 010A 0074"
```

```
$"0110 0074 0112 007   13 0074 0114 0074 0115 0074 0116 0074 0116 0074 0116 0074"
$"0116 0074 0115 007   15 0074 0115 0075 0112 0075 0111 0076 010D 0076 010B 0076"
$"0109 0076 0104 007   FF 0074 00FB 0073 00F9 0072 00F6 0072 00F4 0072 00F0 0073"
$"00EF 0074 00EE 007   EE 0078 00ED 0079 00EE 007C 00F0 007D 00F1 007E 00F2 007F"
$"00F5 0080 00F9 008   FD 0080 0100 007F 0105 007D 0110 007B 0116 007B 0117 007A"
$"011A 007A 011C 007   1E 007A 011F 007A 011F 007C 0120 007E 011F 007E 011F 0080"
$"011E 0083 011B 008  117 008B 0112 008D 010F 008E 010E 0090 010B 0092 0109 0094"
$"0107 0096 0106 009  105 009A 0104 009C 0104 009D 0104 009F 0104 00A3 0104 00A7"
$"0103 00AB 0102 00A  101 00AF 0101 00B0 0100 00B3 00FE 00B3 00FD 00B4 00FC 00B4"
$"00FA 00B4 00F6 00B   F5 00B3 00F3 00B2 00F1 00B0 00EC 00AE 00E7 00AD 00E3 00AC"
$"00E0 00AC 00DF 00A   DC 00AA 00DB 00AA 00DA 00A9 00D9 00A7 00D8 00A7 00D8 00A6"
$"00D8 00A5 00D8 00A   D8 00A1 00D8 009D 00D9 009A 00DA 0098 00DA 0094 00DB 008C"
$"00DB 0088 00DB 003   DB 0082 00DB 007C 00DA 0077 00DA 0074 00D9 006F 00D8 0064"
$"00D6 005F 00D5 005   D5 0051 00D5 004C 00D5 004A 00D5 0045 00D6 0040 00D7 003B"
$"00D8 0038 00D9 003   DA 0031 00DC 002B 00DF 0027 00E2 0025 00E3 0022 00E5 001E"
$"00E8 001B 00EC 001   EF 0017 00F1 0016 00F3 0013 00F7 0011 00FB 0010 00FF 000F"
$"0102 000E 0106 000   0E 000B 0113 0100 0A00 0000 0000 0000 0007 0001 0001 2200"
$"0B01 14F0 0623 F90   F8 0623 FD0A 2307 FD23 04FC 2302 0223 F311 23FD 1323 0113"
$"2307 0F23 0B09 230   23 0B00 23FB 0123 EF02 23EE F923 FC09 230A 0823 14FD 2318"
$"F923 F90E 23E8 102   10 23FA 1223 EBF7 23EE FC23 01F6 2303 F023 00F0 23FC E923"
$"FDEC 2301 EE23 08E   0C EF23 10F5 2311 FB23 09FF 0A88 2288 2288 2288 2284 000A"
$"0000 0000 0000 000   00 A301 000A FFFF FFFF 011E 01FC 8400 0A00 0000 0000 0000"
$"0022 000B 0114 FE0   FD 0123 FD01 23FC 0223 FD01 23FF 0023 FE01 23FE 0123 0000"
$"23FE 0123 FC03 23F   23 FE02 23FD 0423 FF02 2300 0123 0001 2300 0123 0101 2301"
$"FF23 0200 2303 FE2   FF 2300 0023 01FF 2301 0023 0100 2300 0023 0000 2300 0123"
$"FF02 23FE 0423 FE0   FE 0323 FE03 23FE 0523 FE05 2300 0223 0002 23FF 0523 0005"
$"2300 0523 0002 230   23 0105 2301 0423 0104 2301 0223 0102 2302 0323 0203 2303"
$"0323 0101 2303 022   04 2303 0123 0301 2306 0123 0200 2301 0023 0100 2301 0023"
$"0100 2300 0023 000   00 0023 FF00 2300 0023 0000 23FD 0123 FF00 23FC 0123 FE00"
$"23FE 0023 FB00 23F   23 FCFF 23FE FF23 FDFF 23FE 0023 FC00 23FF 0123 FF01 2300"
$"0123 FF03 2301 012   03 2301 0123 0101 2303 0123 0401 2304 0023 0300 2305 FF23"
$"0BFE 2306 FE23 011  303 FF23 0200 2302 0023 0100 2300 0023 0102 23FF 0223 0000"
$"23FF 0223 FD03 23F  423 FB04 23FD 0223 FF01 23FD 0223 FE02 23FE 0223 FF02 23FF"
$"0223 FF02 2300 022   01 2300 0223 0004 23FF 0423 FF04 23FF 0323 0001 23FF 0123"
$"FE03 23FF 0023 FFC   FE 0023 FC00 23FF 0023 FEFF 23FE FF23 FBFE 23FB FE23 FCFF"
$"23FD FF23 FF00 23F   23 FFFF 23FF 0023 FFFF 23FF FE23 0000 2300 FF23 00FF 2300"
$"FF23 00FD 2301 FC2   FD 2300 FE23 01FC 2300 F823 00FC 2300 FE23 00FC 23FF FA23"
$"00FB 23FF FD23 FFF  3FE F523 FFFB 2300 FB23 00F7 2300 FB23 00FE 2301 FB23 01FB"
$"2301 FB23 01FD 230   23 02FC 2303 FA23 03FC 2301 FE23 02FD 2303 FC23 04FD 2303"
$"FD23 02FF 2302 FF2   FD 2304 FE23 04FF 2303 FF23 04FF 2308 FE23 05FF A000 A1A1"
$"00B6 0004 0001 000   00 0000 0023 0000 2200 9600 D900 00A0 00A0 A100 A400 0201"
$"0101 000A 0000 000   00 0000 0700 0100 0109 FFFF FFFF FFFF FFFF 2200 AD00 D405"
$"E923 00E9 23F8 DC2   E2 2312 DE23 20EE 2325 0323 1411 230A 1A23 021A 23F4 2423"
$"E917 23E6 1823 F11   00 A301 000A FFFF FFFF 011E 01FC 2200 AD00 D401 FD23 01FA"
$"2301 FB23 00FD 230   23 01FC 2300 FC23 00FD 2300 FF23 00FD 2300 FA23 FFF8 23FE"
$"F823 FFFB 23FF FC2   F7 23FF F823 00F8 2300 FC23 00FD 2302 F723 02F9 2304 F723"
$"02FC 2302 FC23 06F   06 F923 07FB 2304 FE23 02FF 2304 FE23 05FE 2304 FF23 04FF"
$"2303 0023 0500 230    3 0200 2302 0023  501 2304 0023 0301 2305 0223 0201 2304"
$"0223 0302 2301 012   02 2304 0523 0405 2303 0623 0103 2301 0323 0206 2302 0823"
$"0106 2300 0323 000   00 0723 FF08 23FE 0823 FE05 23FF 0423 FC08 23FC 0723 FB07"
$"23FD 0323 FA06 23F   23 F906 23FE 0223 FD03 23FD 0323 FC05 23FD 0323 FC06 23FD"
$"0423 FC06 23FE 03A   A1 A100 B600 0400 4000 80A1 00D8 0004 0000 4000 A100 D600"
$"0400 0040 00A0 00D   00 0000 0008 0017 23EB F9A1 00CA 0008 0000 0000 0000 0000"
$"A100 C800 0800 000   00 1300 1651 00B0 00D3 00BA 0100 0700 0100 0158 A000 C9A1"
$"00D6 0004 0000 40    00 D7A0 00D9 A100 D600 0400 0040 00A1 00D8 0004 0000 4000"
$"A000 BEA0 00D9 08     1 00A6 00AC 00D5 00BC 00FE 00AD 00D5 00AD 00D5 00AC 00D5"
$"00AC 00D7 00AC 00    AC 00DC 00AC 00DD 00AD 00E1 00AE 00E5 00AF 00E8 00B0 00EB"
$"00B2 00EF 00B5 00F   B6 00F9 00B7 00FA 00B9 00FD 00BA 00FE 00BB 00FE 00BB 00FE"
$"00BB 00FE 00BC 00F   BC 00FB 00BC 00F8 00BC 00F7 00BC 00F5 00BC 00F2 00BB 00EE"
$"00BA 00EA 00B9 00E   B3 00E5 00B7 00E2 00B5 00DF 00B3 00DB 00B2 00DA 00B1 00D8"
$"00B0 00D7 00AF 00D   AE 00D5 00AD 00D5 7000 A600 AC00 D500 BC00 FE00 AD00 D500"
$"AD00 D500 AC00 D50   00 D700 AC00 DB00 AC00 DC00 AC00 DD00 AD00 E100 AE00 E500"
$"AF00 E800 B000 EB0   00 EF00 B500 F600 B600 F900 B700 FA00 B900 FD00 BA00 FE00"
$"BB00 FE00 BB00 FE0   00 FE00 BC00 FD00 BC00 FB00 BC00 F800 BC00 F700 BC00 F500"
```

```
$"BC00 F200 BB00 EE00 BA00 EA00 B900 E800 B800 E500 B700 E200 B500 DF00 B300 DB00"
$"B200 DA00 B100 D800 B000 D700 AF00 D600 AE00 D500 AD00 D5A1 00D8 0004 0000 4000"
$"A000 BFA0 00D9 A000 00A1 00B6 0004 0001 0001 0700 0000 0022 00AD 00D5 0000 A100"
$"B600 0400 4000 8022 000D 011D 0000 A000 A0A1 00A4 0002 0340 0988 2288 2288 2288"
$"2271 0236 000E 00F1 0370 0144 000F 0129 000F 0126 000E 0121 000E 011D 000E 0119"
$"000E 0118 000F 0115 0011 0110 0012 010E 0013 010C 0016 0107 0018 0105 0019 0103"
$"001B 0101 001D 00FF 0020 00FD 0021 00FC 0022 00FB 0026 00F9 0029 00F7 002C 00F6"
$"002F 00F5 0031 00F4 0035 00F3 0039 00F2 003D 00F2 003F 00F2 0041 00F2 0045 00F1"
$"0048 00F1 004B 00F1 004C 00F1 004D 00F1 0050 00F2 0052 00F2 0056 00F3 0057 00F3"
$"0057 00F3 0059 00F4 0059 00F4 0058 00F4 0057 00F4 0055 00F4 0051 00F4 004E 00F4"
$"004C 00F4 0048 00F4 0041 00F4 003A 00F5 0035 00F6 0033 00F6 0030 00F7 002C 00F8"
$"0028 00FA 0024 00FC 0023 00FD 0020 0100 001B 0105 0018 0108 0017 0109 0015 010D"
$"0014 010F 0013 0113 0012 0115 0011 0117 0010 011B 000F 011E 000F 0121 000F 0123"
$"000F 0126 0010 012C 0011 012E 0013 0131 0016 0135 0018 0137 0019 0138 001D 013B"
$"0022 013D 0026 013E 002A 013F 002B 013F 002E 0140 0032 0141 0034 0141 0036 0141"
$"003A 0142 003B 0142 003C 0142 003D 0142 0040 0142 0046 0141 0049 0140 004B 0140"
$"0051 013E 0054 013C 0057 013B 005C 0139 005E 0138 0060 0137 0064 0134 0066 0132"
$"0068 0131 006D 012E 006E 012D 006E 012D 006F 012C 0070 012C 006F 012C 006F 012C"
$"006D 012E 006D 012E 006C 012F 006B 0130 0069 0132 0066 0134 0063 0136 0060 0138"
$"005E 0139 005D 013A 0059 013C 0055 013E 0052 013F 004F 0140 004A 0142 0042 0144"
$"003F 0144 003E 0144 003A 0144 0035 0144 0032 0144 0030 0143 002C 0142 0025 013F"
$"0021 013D 001F 013C 001C 0139 0018 0135 0013 0130 0011 012D 0100 0A00 0000 0000"
$"0000 0007 0001 0001 2200 0F01 29F4 FE23 F502 23F7 0523 F707 23F8 0B23 FA11 23FF"
$"1023 0109 2303 0D23 FFDF 2304 EE23 0AF3 230B F723 0FFB 230D 0023 0A05 2309 0923"
$"061B 2300 0B23 FD0B 23FB 0C23 FC08 23F9 0923 FC06 230B F323 07F4 2306 EE23 02F2"
$"23FC F023 F8F2 23F4 F784 000A 0000 0000 0000 0000 A000 A301 000A FFFF FFFF 011E"
$"01FC 8400 0A00 0000 0000 0000 0022 000F 0129 FD00 23FB FF23 FC00 23FC 0023 FF00"
$"23FD 0123 FB02 23FE 0123 FE01 23FB 0323 FE02 23FE 0123 FE02 23FE 0223 FE03 23FF"
$"0123 FF01 23FE 0423 FE03 23FF 0323 FF03 23FF 0223 FF04 23FF 0423 0004 2300 0223"
$"0002 23FF 0423 0001 2300 0323 0001 2300 0123 0103 2300 0223 0104 2300 0123 0000"
$"2301 0223 0000 2300 FF23 00FF 2300 FE23 00FC 2300 FD23 00FE 2300 FC23 00F9 2301"
$"F923 01FB 2300 FE23 01FD 2301 FC23 02FC 2302 FC23 01FF 2303 FD23 05FB 2303 FD23"
$"01FF 2304 FE23 02FE 2304 FF23 02FF 2302 FF23 04FF 2303 FF23 0300 2302 0023 0300"
$"2306 0123 0201 2303 0223 0403 2302 0223 0101 2303 0423 0205 2301 0423 0104 2300"
$"0123 0103 2301 0423 0002 2300 0223 0104 2300 0123 0001 2300 0123 0003 23FF 0623"
$"FF03 2300 0223 FE04 23FE 0323 FF03 23FE 0523 FF02 23FF 0223 FD04 23FE 0223 FF02"
$"23FD 0523 FF01 2300 0023 FF01 2300 0123 00FF 2300 0023 02FE 2300 0023 01FF 2301"
$"FF23 02FE 2302 FD23 02FD 2302 FD23 01FE 2301 FF23 02FC 2302 FC23 01FD 2301 FD23"
$"02FB 2302 F823 00FE 2300 FF23 00FC 2300 FB23 00FD 23FF FE23 FFFC 23FD F923 FEFC"
$"23FF FE23 FDFD 23FC 0023 FBFB 23FD FEA0 00A1 A100 B600 0400 0100 0107 0000 0000"
$"2300 00A1 00B6 0004 0040 0080 2200 6901 3A00 00A0 00A0 A100 A400 0207 4009 AA55"
$"AA55 AA55 AA55 7101 4A00 4B00 DC00 B601 4800 6101 3F00 6301 3E00 6701 3B00 6B01"
$"3800 6F01 3400 7101 3200 7301 3000 7701 2B00 7B01 2600 8001 1F00 8201 1C00 8501"
$"1800 8901 1000 8C01 0900 8E01 0100 8E00 FE00 8F00 FA00 8F00 F300 8F00 EE00 8E00"
$"E800 8D00 E500 8D00 E400 8C00 E200 8C00 E000 8C00 DE00 8C00 DD00 8C00 DD00 8D00"
$"DD00 9000 DD00 9000 DD00 9100 DD00 9400 DD00 9900 DD00 9C00 DD00 9F00 DC00 A000"
$"DC00 A200 DC00 A500 DC00 A500 DC00 A700 DD00 A800 DE00 AA00 E000 AB00 E200 AB00"
$"E300 AD00 E800 B100 F200 B300 F600 B400 F800 B600 FC00 B600 FD00 B600 FF00 B501"
$"0000 B501 0000 B301 0100 B001 0300 AD01 0500 AC01 0600 AB01 0700 A701 0A00 A401"
$"0D00 9F01 1200 9D01 1400 9C01 1500 9A01 1700 9801 1800 9701 1900 9601 1900 9501"
$"1800 9501 1700 9601 1600 9601 1500 9701 1300 9701 1000 9801 0C00 9801 0900 9801"
$"0800 9801 0700 9801 0500 9801 0500 9701 0500 9601 0500 9601 0500 9401 0600 9401"
$"0700 9301 0A00 9301 0900 9301 0C00 9201 0F00 9001 1400 8E01 1900 8D01 1B00 8B01"
$"2000 8501 2A00 8101 2F00 7D01 3300 7501 3A00 7001 3D00 6E01 3E00 6A01 4000 6601"
$"4200 6001 4400 5E01 4400 5B01 4500 5801 4500 5401 4600 5001 4700 4F01 4700 4F01"
$"4700 4E01 4700 4D01 4700 4B01 4800 4C01 4800 4B01 4800 4C01 4700 4D01 4700 4D01"
$"4700 4F01 4700 5301 4600 5501 4500 5501 4500 5801 4300 5B01 4200 5F01 4000 6101"
$"3F01 000A 0000 0000 0000 0000 0700 0100 0122 0058 0143 F711 23F1 0F23 E214 23E2"
$"0423 ECF9 2303 0D23 0112 2313 0623 110A 2303 FA23 09F5 2314 EC23 EF05 23F7 0123"
$"01FB 230B FE23 16FF 0111 F123 0BEF 2304 ED23 02F4 23FE 0823 FD07 0AAA 55AA 55AA"
$"55AA 5584 000A 0000 0000 0000 0000 A000 A301 000A FFFF FFFF 011E 01FC 8400 0A00"
$"0000 0000 0000 0020 0061 013F FF02 23FD 0423 FD04 23FC 0423 FE02 23FE 0223 FB04"
$"23FB 0423 F905 23FB 0223 FC03 23F8 0423 F903 23F8 0223 FD00 23FC 0123 F900 23FB"
$"0023 FAFF 23FD FF23 FF00 23FE FF23 FE00 23FE 0023 FF00 2300 0023 0001 2300 0323"
```

```
$"0000 2300 0123 02   2300 0523 0003 23FF 0323 0001 2300 0223 0003 2300 0023 0102"
$"2301 0123 0202 23    123 0100 2305 0223 0A04 2304 0223 0201 2304 0223 0100 2302"
$"0023 01FF 2300 02    1FE 2302 FD23 02FD 2301 FF23 01FF 2303 FC23 03FD 2305 FB23"
$"02FE 2301 FF23 02    301 FE23 01FF 2300 FF23 FFFF 23FF 0023 FF01 23FF 0023 FE01"
$"23FD 0023 FC01 23    023 FF00 23FF 0023 FE00 2300 0023 00FF 2300 FF23 0000 2301"
$"FE23 0100 2303 FF    100 2301 0023 03FF 2305 FE23 05FE 2302 FF23 05FE 230A FA23"
$"05FC 2304 FC23 02    303 FB23 01FE 2302 FC23 02FC 2302 FA23 00FE 2301 FD23 00FD"
$"2301 FC23 01FC 23    F23 0000 2300 FF23 00FF 2301 FE23 0001 2300 FF23 FF01 2300"
$"0123 0000 2300 022   F04 23FF 0223 0000 23FE 0323 FF03 23FE 0423 FF02 A000 A1A1"
$"00B6 0004 0001 00    700 0000 0023 0000 A100 B600 0400 4000 8022 0055 00D4 0000"
$"A000 A0A1 00A4 00    140 0100 0A00 0000 0000 0000 0007 0001 0001 09FF FFFF FFFF"
$"FFFF FF22 004D 00D   208 230B 0923 12FC 2304 F923 0411 230C 0D23 1003 230F FC23"
$"07FC 23FD 0C23 060   305 01A0 00A3 0100 0AFF FFFF FF01 1E01 FC22 004D 00D2 0102"
$"2301 0423 0203 230   023 0101 2301 0123 0301 2304 0123 0300 2303 FF23 0200 2304"
$"FE23 03FF 2301 FF2   1FF 2300 0023 01FE 2300 0123 0100 2300 0023 0102 2300 0123"
$"0102 2300 0123 010   001 0423 0204 2303 0423 0101 2302 0223 0302 2303 0223 0402"
$"2302 0023 0200 230   123 0400 2304 0023 02FF 2301 0023 03FF 2303 FF23 03FF 2301"
$"FF23 0000 2301 002   100 2300 0323 0001 2300 0123 FF04 2300 0123 0102 2301 0123"
$"0101 2302 0223 022   002 0123 0100 A000 A1A1 00B6 0004 0001 0001 0700 0000 0023"
$"0000 2200 D800 C6    1A0 00A0 A100 A400 020B 0109 77DD 77DD 77DD 77DD 7100 4200"
$"BE00 C600 E800 F8    D00 CF00 D800 C600 DB00 C700 DE00 CA00 E000 CE00 E200 D200"
$"E400 D500 E500 D9    100 DE00 E800 E300 E800 E800 E800 EC00 E700 EF00 CC00 F801"
$"000A 0000 0000 000   000 0700 0100 0109 FFFF FFFF FFFF FFFF 2200 BE00 CFF7 1A23"
$"0000 2301 0323 00    303 0323 0000 2304 0223 0000 2304 0223 0000 2303 0223 0000"
$"2304 0123 0000 23    023 0000 2305 0123 0000 2305 0023 0000 2304 0023 0000 2303"
$"FF23 0000 2309 E52   100 0A77 DD77 DD77 DD77 DD84 000A 0000 0000 0000 0000 A000"
$"A301 000A FFFF FF    11E 01FC 8400 0A00 0000 0000 0000 0022 00BE 00CF F71A 2301"
$"0323 0303 2304 02    102 2303 0223 0401 2305 0223 0501 2305 0023 0400 2303 FF23"
$"09E5 A000 A1A1 00    004 0000 4000 A100 D600 0400 0040 00A0 00D7 0700 0000 0008"
$"0017 23EC FAA1 00    008 0000 0000 0000 0000 A100 C800 0800 0000 1300 1300 1651"
$"00C3 00CE 00CA 00    100 0100 0158 A000 C9A1 0CD6 0004 0000 4000 A000 D7A0 00D9"
$"A100 D600 0400 00    0A1 00D8 0004 0000 4000 A000 BEA0 00D9 0800 0871 009E 00BF"
$"00CF 00CE 00F8 00    0CF 00BF 00CF 00BF 00D0 00BF 00D2 00BF 00D5 00BF 00D6 00BF"
$"00D7 00C0 00DB 00    0DF 00C2 00E2 00C3 00E5 00C5 00E9 00C8 00F0 00C9 00F3 00CA"
$"00F5 00CB 00F7 00    0F8 00CC 00F8 00CD 00F8 00CD 00F8 00CE 00F7 00CE 00F6 00CD"
$"00F2 00CD 00F1 00    0EE 00CB 00E7 00C9 00E3 00C8 00E0 00C7 00DD 00C5 00D8 00C4"
$"00D6 00C3 00D4 00    0D2 00C1 00D1 00C0 00D0 00BF 00CF 00BF 00CF 7000 9E00 BF00"
$"CF00 CE00 F800 BF    F00 BF00 CF00 BF00 D000 BF00 D200 BF00 D500 BF00 D600 BF00"
$"D700 C000 DB00 C1    F00 C200 E200 C300 E500 C500 E900 C800 F000 C900 F300 CA00"
$"F500 CB00 F700 CC    800 CC00 F800 CD00 F800 CD00 F800 CE00 F700 CE00 F600 CD00"
$"F200 CD00 F100 CD    E00 CB00 E700 C900 E300 C800 E000 C700 DD00 C500 D800 C400"
$"D600 C300 D400 C2    200 C100 D100 C000 D000 BF00 CF00 BF00 CFA1 00D8 0004 0000"
$"4000 A000 BFA0 00    000 D7A1 00B6 0004 0040 0080 0700 0000 0022 00BF 00C2 0000"
$"A000 A0A1 00A4 00    340 7101 AA00 C100 7B00 F300 CC00 C500 C900 C400 C800 C200"
$"C400 C100 C100 C1    E00 C100 BC00 C200 BA00 C300 B800 C500 B600 C800 B300 CA00"
$"B200 CE00 B000 D4    D00 D800 A900 D900 A800 DA00 A700 DC00 A500 DC00 A400 DD00"
$"A300 DD00 A100 DD    F00 DD00 9C00 DD00 9B00 DD00 9900 DD00 9600 DD00 9200 DE00"
$"8E00 DF00 8C00 E0    A00 E200 8600 E500 8300 E800 8100 EB00 7F00 ED00 7E00 ED00"
$"7D00 EF00 7C00 F1    B00 F000 7B00 F100 7B00 F000 7C00 F000 7C00 EF00 7D00 EE00"
$"7E00 ED00 7F00 ED    F00 ED00 8000 ED00 8100 EE00 8100 EE00 8100 F000 8000 F000"
$"8000 F000 8000 F2    F00 F300 7F00 F300 7F00 F300 7F00 F200 8000 F200 8000 F000"
$"8100 F000 8100 EF    200 EC00 8400 EA00 8600 E800 8800 E700 8900 E600 8B00 E500"
$"8F00 E500 9100 E6    300 E600 9800 E600 9A00 E600 9B00 E600 9D00 E500 A000 E300"
$"A400 E200 A500 E0    900 DB00 AD00 D800 AF00 D500 B100 D000 B400 CE00 B500 CD00"
$"B500 CC00 B600 CA      0 C900 B900 C800 BA00 C700 BB00 C600 BD00 C500 BF00 C500"
$"C000 C500 C100 C5      0 C600 C800 C700 CA00 C700 CA00 C800 CC00 C800 CB00 C800"
$"CC00 C800 CC00 C7      0 C700 CB00 C600 CA00 C500 C901 000A 0000 0000 0000 0000"
$"0700 0100 0122 00     09 F9FA 23F4 0423 F80E 23F6 0E23 EFFD 23F2 0523 F313 230A"
$"F623 FC0C 2307 F4     FB 2308 0123 0A00 230C F823 08F4 2303 F823 07FB 2308 0023"
$"0905 23FB FB0A FF     FF FFFF FFFF 8400 0A00 0000 0000 0000 00A0 00A3 0100 0AFF"
$"FFFF FF01 1E01 FC     0A 0000 0000 0000 0000 23FF FF23 FCFE 23FD FF23 FD00 23FE"
$"0023 FE01 23FE 01     02 23FD 0323 FF02 23FE 0423 FC06 23FD 0423 FF01 23FF 0123"
$"FE02 23FF 0023 FF     FE 0023 FE00 23FD 0023 FF00 23FE 0023 FD00 23FC 0023 FC01"
$"23FE 0123 FE01 23     23 FD03 23FE 0323 FE03 23FF 0223 FF00 23FF 0223 FF02 2300"
```

```
$"FF23 0001 2301 FF23 0000 2301 FF23 01FF 2301 FF23 0000 2301 0023 0100 2300 0123"
$"0000 23FF 0223 0000 2300 0023 FF02 2300 0123 0000 2300 0023 01FF 2300 0023 01FE"
$"2300 0023 01FF 2302 FD23 02FE 2302 FE23 01FF 2302 FF23 04FF 2302 0023 0201 2305"
$"0023 0200 2301 0023 0200 2303 FF23 04FE 2301 FF23 03FE 2305 FB23 02FD 2302 FD23"
$"03FB 2301 FE23 00FF 2301 FF23 01FE 2302 FF23 01FF 2301 FF23 02FF 2302 FF23 0100"
$"2301 0023 0200 2303 0123 0201 2300 0023 0201 23FF 0023 0100 2300 0023 FFFF 2300"
$"0023 FFFF 23FF FFA0  0A1 A100 B600 0400 0100 0107 0000 0000 2300 00A1 00B6 0004"
$"0040 0080 2200 1201  000 00A0 00A0 A100 A400 0203 4009 8000 0800 8000 0800 7101"
$"A200 1100 F800 6F01  800 1101 2500 1101 2200 1201 1D00 1301 1800 1401 1500 1501"
$"1300 1601 1100 1801  000 1A01 0B00 1D01 0700 1F01 0500 2101 0300 2501 0000 2700"
$"FE00 2B00 FC00 2D00  300 2F00 FA00 3200 F900 3600 F800 3B00 F800 3D00 F800 4200"
$"F800 4C00 F800 5100   00 5300 F800 5700 F800 5B00 F900 5E00 FA00 6000 FB00 6200"
$"FC00 6400 FE00 6701   00 6901 0300 6A01 0400 6B01 0600 6D01 0900 6E01 0D00 6F01"
$"1000 6F01 1200 6F0    00 6F01 1800 6F01 1D00 6F01 2000 6E01 2200 6E01 2300 6C01"
$"2700 6B01 2900 690   000 6801 2D00 6601 2F00 6101 3300 5F01 3400 5D01 3500 5701"
$"3700 5301 3800 4F0    00 4701 3B00 4401 3B00 4201 3B00 3F01 3B00 3C01 3A00 3A01"
$"3800 3901 3700 380    00 3801 3400 3801 3200 3801 2F00 3901 2E00 3A01 2C00 3C01"
$"2700 3D01 2400 3E01   00 3E01 2000 3E01 1D00 3D01 1B00 3C01 1A00 3B01 1900 3801"
$"1700 3501 1500 3101   00 3001 1400 2F01 1400 2B01 1400 2801 1500 2601 1600 2501"
$"1700 2301 1900 2001  000 1F01 1F00 1E01 2200 1D01 2700 1D01 2A00 1D01 2B00 1C01"
$"2D00 1B01 2E00 1A0   F00 1901 2F00 1801 2F00 1701 2F00 1501 2E00 1301 2C00 1201"
$"2B01 000A 0000 0000  000 0000 0700 0100 0122 0011 0125 F401 23F4 0623 F10E 23FA"
$"0D23 FF14 2301 132   50B 230C 0923 1102 230E FC23 08F8 2305 F623 04F1 2301 F123"
$"F7FA 23F8 0523 F40   3F7 F723 FEF2 2307 F823 0AFC 230B FF23 00FA 23FC FC0A 8000"
$"0800 8000 0800 840   A00 0000 0000 0000 00A0 00A3 0100 0AFF FFFF FF01 1E01 FC84"
$"000A 0000 0000 000   000 2200 1101 25FD 0023 FB01 23FB 0123 FD01 23FE 0123 FE01"
$"23FD 0223 FD02 23F   323 FE02 23FE 0223 FD04 23FE 0223 FE04 23FF 0223 FF02 23FF"
$"0323 FF04 2300 052   002 2300 0523 000A 2300 0523 0002 2300 0423 0104 2301 0323"
$"0102 2301 0223 020   302 0323 0302 2301 0123 0201 2303 0223 0401 2303 0123 0200"
$"2302 0023 0400 230   023 0300 2302 FF23 0100 2304 FE23 02FF 2303 FE23 01FF 2302"
$"FE23 04FB 2301 FE2   1FE 2302 FA23 01FC 2301 FC23 02F8 2300 FD23 00FE 2300 FD23"
$"FFFD 23FE FE23 FFF    FE FF23 FF00 23FE 0023 FD00 23FF 0123 FE01 23FB 0223 FD01"
$"23FE 0123 FE00 23F    23 FEFF 23FF FF23 FFFF 23FE FD23 FEFD 23FF FC23 00FF 2300"
$"FF23 00FC 2301 FD2   1FE 2301 FF23 02FE 2304 FD23 02FF 2303 FF23 05FF 2303 0023"
$"0100 2302 FF23 01F   301 FF23 00FF 2300 FF23 00FF 23FF FE23 FEFE 23FF FFA0 00A1"
$"A100 B600 0400 010   107 0000 0000 2300 0022 002B 0141 0000 A000 A0A1 00A4 0002"
$"0101 0100 0A00 00C    00 0000 0007 0001 0001 09FF FFFF FFFF FFFF FF22 001B 013A"
$"0710 2300 1923 F61    F1 1423 EC17 23F0 1323 F710 23FC 09A0 00A3 0100 0AFF FFFF"
$"FF01 1E01 FC22 001    3A 0204 2302 0723 0207 2301 0823 0003 2300 0323 FF07 23FF"
$"0623 FE06 23FF 032    03 23FD 0623 FD06 23FC 0623 FE02 23FE 0323 FC05 23FC 0623"
$"FC05 23FD 0323 FB0   3F7 0A23 FC05 23FE 0223 FC05 23FD 0423 FD04 23FF 0223 FE04"
$"23FC 0923 FE04 A00   107 0000 0000 2200 0E01 1D00 00A0 00A0 A100 A400 0201 0101"
$"000A 0000 0000 000    00 0700 0100 0122 000F 012A F3FF 23F1 0623 EB13 23F9 1F23"
$"051B 23FB 2323 FC1   3FA 12A0 00A3 0100 0AFF FFFF FF01 1E01 FC22 000F 012A FD00"
$"23FA 0023 FB00 23F   123 FE01 23FE 0123 FC02 23FC 0323 FB04 23FE 0223 FD03 23FC"
$"0523 FC07 23FE 072    04 23FF 0423 FF07 2300 0723 0007 2301 0423 0104 2300 0723"
$"0007 23FF 0823 FF0    00 0423 FF08 23FF 0723 FF07 23FF 0423 0003 23FE 0723 FE07"
$"23FE 0823 FE05 A00    A1 00D8 0004 0000 4000 A100 D600 0400 0040 00A0 00D7 0700"
$"0000 0008 0017 23F    A1 00CA 0008 0000 0000 0000 0000 A100 C800 0800 0000 1300"
$"1300 1651 00C1 00C    C9 00FB 0700 0100 0109 0000 0000 0000 0000 58A0 00C9 A100"
$"D600 0400 0040 00A    07 A000 D9A1 00D6 0004 0000 4000 A100 D800 0400 0040 00A0"
$"00BE A000 D908 00C    FF FFFF FFFF FFFF FF71 009E 00BD 00D0 00CD 00F9 00BE 00D0"
$"00BE 00D0 00BD 000    BD 00D2 00BD 00D6 00BD 00D7 00BE 00DA 00C1 00E1 00C2 00E5"
$"00C3 00E8 00C4 00E    C5 00EF 00C7 00F3 00C7 00F4 00C8 00F6 00C9 00F7 00CA 00F8"
$"00CC 00F9 00CC 00F    CC 00F9 00CD 00F7 00CD 00F6 00CC 00F3 00CC 00F2 00CC 00F1"
$"00CB 00ED 00CA 00E    C9 00E6 00C8 00E3 00C7 00DF 00C4 00D8 00C2 00D5 00C1 00D3"
$"00C0 00D1 00BF 00C    BF 00D0 00BE 00D0 0900 0000 0000 0000 0070 009E 00BD 00D0"
$"00CD 00F9 00BE 00D    BE 00D0 00BD 00D0 00BD 00D2 00BD 00D6 00BD 00D7 00BE 00DA"
$"00C1 00E1 00C2 00E    C3 00E8 00C4 00EB 00C5 00EF 00C7 00F3 00C7 00F4 00C8 00F6"
$"00C9 00F7 00CA 00F    CC 00F9 00CC 00F9 00CC 00F9 00CD 00F7 00CD 00F6 00CC 00F3"
$"00CC 00F2 00CC 00F    CB 00ED 00CA 00E9 00C9 00E6 00C8 00E3 00C7 00DF 00C4 00D8"
$"00C2 00D5 00C1 00D    C0 00D1 00BF 00D0 00BF 00D0 00BE 00D0 A100 D800 0400 0040"
$"00A0 00BF A000 D9A    07 A100 B600 0400 4000 8007 0000 0000 2200 CF00 BD00 00A0"
$"00A0 A100 A400 0201   01 000A 0000 0000 0000 0000 0700 0100 0109 FFFF FFFF FFFF"
```

```
$"FFFF 2200 0E00 00      23 0210 23F9 0623 F6FC 23EC F423 DEFB 23F6 0E23 020E 23F7"
$"0423 F901 23FC 061     FD 23F9 07A0 00A3 0100 0AFF FFFF FF01 1E01 FC22 00CE 00C9"
$"FF00 23FC 0123 FF      FE 0123 FF01 23FF 0123 FF02 2300 0123 0001 2300 0223 0004"
$"2300 0223 FF02 231     23 FF01 23FD 0123 FE00 23FF 0023 FEFF 23FF 0023 FDFE 23FC"
$"FE23 FCFE 23FD FE.     FF 23FA FD23 F9FE 23F9 FF23 FBFF 23FE 0023 FCFF 23FC 0023"
$"FE00 23FC 0123 FE      FE 0123 FF01 23FF 0123 FF01 23FE 0423 FF04 2300 0323 0002"
$"2300 0123 0004 23E     23 FF02 23FF 0023 FE01 23FB 0223 FF00 23FF 0023 FF00 23FE"
$"0123 FF01 23FF 011     00 23FE 0223 FE00 23FD 0023 FF0. 2300 0023 FE00 23FD 0023"
$"FE00 23FF 0123 FF      FD 0223 FF01 23FF 01A0 00A1 A100 B600 0400 0100 0107 0000"
$"0000 2300 00A0 00      00 B600 0400 0C00 8007 0001 0001 2000 0801 0000 0800 22A1"
$"00B6 0004 0001 00      00 0000 0023 0000 A100 B600 0400 0C00 8007 0001 0001 2000"
$"1100 EA00 1100 30      B6 0004 0001 0001 0700 0000 0023 0000 A100 B600 0400 0C00"
$"8007 0001 0001 22      00 DF00 1B00 41A1 00B6 0004 0001 0001 0700 0000 0023 0000"
$"A100 B600 0400 0C      07 0001 0001 2000 2500 D800 2500 23A1 00B6 0004 0001 0001"
$"0700 0000 0023 00      00 B600 0400 0C00 8007 0001 0001 2200 3200 D28B 00A1 00B6"
$"0004 0001 0001 07      00 0023 0000 A100 B600 0400 0C00 8007 0001 0001 2200 3E00"
$"CD9D 00A1 00B6 00      01 0001 0700 0000 0023 0000 A100 B600 0400 0C00 8007 0001"
$"0001 2200 4900 CA3     A1 00B6 0004 0001 0001 0700 0000 0023 0000 A100 B600 0400"
$"0C00 8007 0001 000     00 5400 CACA 00A1 00B6 0004 0001 0001 0700 0000 0023 0000"
$"A100 B600 0400 0C      07 0001 0001 2200 5E00 CA84 00A1 00B6 0004 0001 0001 0700"
$"0000 0023 0000 A1      00 0400 0C00 8007 0001 0001 2200 6900 CDC2 00A1 00B6 0004"
$"0001 0001 0700 00      23 0000 A100 B600 0400 0C00 8007 0001 0001 2200 7300 D0D3"
$"00A1 00B6 0004 00.     01 0700 0000 0023 0000 A100 B600 0400 0C00 8007 0001 0001"
$"2200 7C00 D1CE 00A     B6 0004 0001 0001 0700 0000 0023 0000 A100 B600 0400 0C00"
$"8007 0001 0001 22      00 D2AB 00A1 00B6 0004 0001 0001 0700 0000 0023 0000 A100"
$"B600 0400 0C00 30      01 0001 2200 9400 D3DA 00A1 00B6 C004 0001 0001 0700 0000"
$"0023 0000 A100 B6      00 0C00 8007 0001 0001 2200 9E00 D0ED 00A1 00B6 0004 0001"
$"0001 0700 0000 002     00 A100 B600 0400 0C00 8007 0001 0001 2200 A800 CEE8 00A1"
$"00B6 0004 0001 00      00 0000 0023 0000 A000 8DA1 0096 000C 0300 0000 0200 0000"
$"0000 0000 A100 9A      00 0400 0000 2D00 0001 000A 00BB 00D1 00EB 01E3 2C00 0E00"
$"210B 4176 616E 74      61 7264 6503 0021 0401 0D00 1828 00CE 012C 0E14 5249 4654"
$"2050 4154 5445 52      04 032B 3518 0E53 696D 756C 6174 696F 6E20 2020 20A0 0097"
$"0100 0AFF FFFF FF      01 FC07 0001 0001 2000 B601 0500 B601 E1A1 00B6 0004 0040"
$"0080 0700 0000 00      C9 00C0 0000 A000 A0A1 00A4 0002 0140 0100 0A00 0000 0000"
$"0000 0007 0001 00      00 C800 CCF4 0123 0210 23F9 0623 F6FC 23EC F423 DEFB 23F6"
$"0E23 020E 23F7 04      01 23FC 0623 F500 23ED FF23 F30D A000 A301 000A FFFF FFFF"
$"011E 01FC 2200 C8      FF 0023 FC01 23FF 0023 FE01 23FF 0123 FF01 23FF 0323 0001"
$"2300 0123 0002 23      23 0002 23FF 0223 FF01 23FF 0123 FD01 23FE 0023 FF00 23FE"
$"FF23 FF00 23FD FE2     FE 23FC FE23 FDFE 23FE FF23 FAFD 23F9 FE23 F9FF 23FB FF23"
$"FE00 23FC FF23 FC0     FE 0023 FC01 23FE 0123 FE01 23FF 0123 FF01 23FF 0123 FE04"
$"23FF 0423 0003 23      23 0001 2300 0423 FF02 23FF 0223 FF00 23FE 0123 FB02 23FF"
$"0023 FF00 23FF 002     01 23FF 0123 FF01 2300 0023 FE02 23FE 0123 FE00 23FF 0023"
$"FF00 23FD 0023 FC0     FB FF23 FE00 23FF 0023 FE00 23FD 0123 FD01 23FD 0223 FD02"
$"23FC 0323 FE02 23F     A0 00A1 A100 B600 0400 0100 0107 0000 0000 2300 00A0 008C"
$"A100 B600 0400 400     07 0001 0001 0900 0000 0000 0000 0020 00D5 01E8 00D5 0147"
$"A100 B600 0400 010     07 0000 0000 2300 00A1 00B6 0004 0040 0080 0700 0100 0120"
$"00D8 01E8 00D8 014     00 B600 0400 0100 0107 0000 0000 2300 00A1 00B6 0004 0040"
$"0080 0700 0100 012     DB 01E8 00DB 0147 A100 B600 0400 0100 0107 0000 0000 2300"
$"00A1 00B6 0004 004     80 0700 0100 0120 00DE 01E8 00DE 0147 A100 B600 0400 0100"
$"0107 0000 0000 230     A1 00B6 0004 0040 0080 0700 0100 0120 00E1 01E8 00E1 0147"
$"A100 B600 0400 010     07 0000 0000 2300 00A1 00B6 0004 0040 0080 0700 0100 0120"
$"00E4 01E8 00E4 014     00 B600 0400 0100 0107 0000 0000 2300 00A1 00B6 0004 0040"
$"0080 0700 0100 012     E6 01E8 00E6 0147 A100 B600 0400 0100 0107 0000 0000 2300"
$"00A1 00B6 0004 004     30 0700 0100 0120 00E9 01E8 00E9 0147 A100 B600 0400 0100"
$"0107 0000 0000 23      A0 008D A100 D800 0400 0040 00A1 00D6 00   000  4000 A000"
$"D708 0017 23A2 CDA     CA 0008 0000 0000 0000 0000 A100 C800 0800 0000 1300 1300"
$"1609 FFFF FFFF FFF     FF 5100 B100 D300 BB00 FF07 0001 0001 0900 0000 0000 0000"
$"0058 A000 C9A1 00C     04 0000 4000 A000 D7A0 00D9 A100 D600 0400 0040 00A1 00D8"
$"0004 0000 4000 A0      A0 00D9 0800 0809 FFFF FFFF FFFF FFFF 7100 9E00 AE00 D400"
$"BE00 FD00 AF00 D40     00 D400 AE00 D500 AE00 D700 AE00 DA00 AE00 DB00 AF00 DF00"
$"B100 E600 B200 EA0     00 ED00 B400 F000 B600 F400 B700 F600 B800 F800 B900 FA00"
$"BA00 FB00 BB00 FC0     00 FD00 BD00 FD00 BE00 FD00 BE00 FD00 BE00 FF00 BE00 F700"
$"BE00 F600 BE00 F50     00 F100 BC00 ED00 BB00 EA00 BA00 E700 B900 E300 B600 DC00"
$"B400 D900 B300 D8      00 D500 B000 D400 B000 D400 AF00 D409 0000 0000 0000 0000"
```

```
$"7000 9E00 AE00 D400 BE00 FD00 AF00 D400 AF00 D400 AE00 D500 AE00 D700 AE00 DA00"
$"AE00 DB00 AF00 DF00 B100 E600 B200 EA00 B300 ED00 B400 F000 B600 F400 B700 F600"
$"B800 F800 B900 FA00 BA00 FB00 BB00 FC00 BC00 FD00 BD00 FD00 BE00 FD00 BE00 FD00"
$"BE00 FB00 BE00 F700 BE00 F600 BE00 F500 BD00 F100 BC00 ED00 BB00 EA00 BA00 E700"
$"B900 E300 B600 DC00 B400 D900 B300 D800 B100 D500 B000 D400 B000 D400 AF00 D4A1"
$"00D8 0004 0000 4000 A000 BFA0 00D9 A000 D7A1 00B6 0004 0040 0080 0700 0000 0022"
$"007B 00E0 0000 A000 A0A1 00A4 0002 0140 0100 0A00 0000 0000 0000 0007 0001 0001"
$"09FF FFFF FFFF FFFF FF12 0058 00D8 0823 2300 1923 FE12 23FD 1023 FE06 23FD 06A0"
$"00A3 0100 0AFF FFFF FF01 1E01 FC22 0058 00D8 0104 2302 0923 0106 2302 0823 0106"
$"2300 0423 0106 2300 0323 0002 2300 0323 0005 2300 0523 FF06 2300 0223 0004 23FE"
$"0923 FF04 2300 0223 FF04 2300 0223 FF03 2300 0023 0000 23FF 0323 FF03 23FF 0323"
$"FF02 A000 A1A1 00B6 0004 0001 0001 0700 0000 0023 0000 A100 B600 0400 4000 8022"
$"0093 00FA 0000 A000 A0A1 00A4 0002 0140 0100 0A00 0000 0000 0000 0007 0001 0001"
$"2200 7201 0AF0 2123 F51E 23F9 1323 FE06 A000 A301 000A FFFF FFFF 011E 01FC 2200"
$"7201 0AFE 0423 FC08 23FD 0723 FD07 23FD 0623 FE05 23FE 0523 FF04 23FF 0223 FF03"
$"23FD 0723 FE06 23FE 0523 FF03 23FF 0323 FE04 23FF 0423 FF03 23FF 02A0 00A1 A100"
$"B600 0400 0100 0107 0000 0000 2300 00A1 00B6 0004 0040 0080 2200 9101 1700 00A0"
$"00A0 A100 A400 0201 4001 000A 0000 0000 0000 0000 0700 0100 0122 007E 012C EB13"
$"22EC 1723 F21B 23FD 0AA0 00A3 0100 0AFF FFFF FF01 1E01 FC22 007E 012C FE02 23FA"
$"0523 FC04 23FC 0423 FB04 23FE 0323 FC04 23FD 0323 FF01 23FE 0323 FB06 23FC 0623"
$"FE06 23FE 0423 FE04 23FD 0623 FE06 23FE 0523 FF03 A000 A1A1 00B6 0004 0001 0001"
$"0700 0000 0023 0000 A100 B600 0400 4000 8022 00B8 00D4 0000 A000 A0A1 00A4 0002"
$"0140 0100 0A00 0000 0000 0000 0007 0001 0001 2200 BF00 D202 F923 01FB 2300 FCA0"
$"00A3 0100 0AFF FFFF FF01 1E01 FC22 00BF 00D2 01FE 2300 FE23 01FE 2301 FD23 0000"
$"2300 FF23 00FC 2300  A0 00A1 A100 B600 0400 0100 0107 0000 0000 2300 00A1 00B6"
$"0004 0040 0080 2200  00 E300 00A0 00A0 A100 A400 0201 4001 000A 0000 0000 0000"
$"0000 0700 0100 0122  0C2 00E1 02FA 2301 FAA0 00A3 0100 0AFF FFFF FF01 1E01 FC22"
$"00C2 00E1 01FD 2302  A23 00FD A000 A1A1 00B6 0004 0001 0001 0700 0000 0023 0000"
$"A100 B600 0400 4000  022 00C6 00F8 0000 A000 A0A1 00A4 0002 0140 0100 0A00 0000"
$"0000 0000 0007 0001  001 2200 CA00 F602 FC23 02FB 2303 FCA0 00A3 0100 0AFF FFFF"
$"FF01 1E01 FC22 00CA  0F6 01FE 2302 FC23 00FF 2301 FF23 02FD 2301 FEA0 00A1 A100"
$"B600 0400 0100 01    000 0000 2300 00A1 00B6 0004 0040 0080 2200 C800 E700 00A0"
$"00A0 A100 A400 020   001 000A 0000 0000 0000 0000 0700 0100 0109 0000 0000 0000"
$"0000 2200 C400 E8FF  423 0000 23FF 0123 0000 A000 A301 000A FFFF FFFF 011E 01FC"
$"2200 C400 E9FF 04    F01 A000 A1A1 00B6 0004 0001 0001 0700 0000 0023 0000 A100"
$"B600 0400 4000 802   0CA 00F3 0000 A000 A0A1 00A4 0002 0940 0100 0A00 0000 0000"
$"0000 0007 0001 000   200 C800 F300 0223 0000 23FF 0123 0000 2300 0123 0000 2300"
$"0023 0000 A000 A30   00A FFFF FFFF 011E 01FC 2200 C800 F300 0223 FF01 2300 0123"
$"0000 A000 A1A1 00B   0004 0001 0001 0700 0000 0023 0000 A100 B600 0400 4000 8007"
$"0001 0001 2200 BF0   8FF 04A1 00B6 0004 0001 0001 0700 0000 0023 0000 A100 B600"
$"0400 4000 8022 00D   0A2 0000 A000 A0A1 00A4 0002 0140 0100 0A00 0000 0000 0000"
$"0007 0001 0001 220   B00 A6FC 0423 FD01 23FC 0123 FBFF 23FB FF23 F901 23FC 0323"
$"FB03 A000 A301 000   FFF FFFF 011E 01FC 2200 DB00 A6FF 0123 FE02 23FF 0123 FF00"
$"2300 0023 FF01 23F   123 FF00 23FF 0023 FE00 23FF 0023 FF00 23FD FF23 FF00 23FF"
$"0023 FC00 23FE 002   F00 23FD 0123 FF01 2300 0023 FE01 23FE 0123 FF01 23FE 01A0"
$"00A1 A100 B600 040   100 0107 0000 0000 2300 00A1 0096 0000 0200 0000 0500 0000"
$"0000 0002 A100 9A0   8FF F800 0000 7C00 0001 000A 008D 00DA 00BE 01FB 2C00 0800"
$"1405 5469 6D65 730   C14 0419 0D00 2828 00AD 00EE 0742 414C 4C4F 4F4E A000 97A1"
$"00B6 0004 000C 008   A100 B600 0400 0100 0101 000A FFFF FFFF 011E 01FC 2200 AD01"
$"E600 00A1 00B6 0004  00C 0080 A100 B600 0400 0100 0123 0000 A100 B600 0400 4000"
$"80A1 00B6 0004 000   201 2300 00A1 00B6 0004 0040 0080 A100 B600 0400 0100 0123"
$"0000 A100 B600 040   000 80A1 00B6 0004 0001 0001 2300 00A1 00B6 0004 0040 0080"
$"A100 B600 0400 01    123 0000 A100 B600 0400 4000 80A1 00B6 0004 0001 0001 2300"
$"00A1 00B6 0004 004   080 A100 B600 0400 0100 0123 0000 A100 B600 0400 4000 80A1"
$"00B6 0004 0001 00    300 00A1 00B6 0004 0040 0080 A100 B600 0400 0100 0123 0000"
$"A100 B600 0400 40    0A1 00B6 0004 0001 0001 2300 00A1 00B6 0004 000C 0080 A100"
$"B600 0400 0100 01    00 A100 B600 0400 0C00 80A1 00B6 0004 0001 0001 2300 00A1"
$"00B6 0004 000C 00    100 B600 0400 0100 0123 0000 A100 B600 0400 0C00 80A1 00B6"
$"0004 0001 0001 23    0A1 00B6 0004 000C 0080 A100 B600 0400 0100 0123 0000 A100"
$"B600 0400 0C00 80    0B6 0004 0001 0001 2300 00A1 00B6 0004 000C 0080 A100 B600"
$"0400 0100 0123 00    100 B600 0400 0C00 80A1 00B6 0004 0001 0001 2300 00A1 00B6"
$"0004 000C 0080 A1    600 0400 0100 0123 0000 A100 B600 0400 0C00 80A1 00B6 0004"
$"0001 0001 2300 00    0B6 0004 000C 0080 A100 B600 0400 0100 0123 0000 A100 B600"
$"0400 0C00 80A1 00    004 0001 0001 2300 00A1 00B6 0004 000C 0080 A100 B600 0400"
```

```
        $"0100 0123 0000 A100 B600 0400 0C00 80A1 00B6 0004 0001 0001 2300 00A1 00B6 0004"
        $"000C 0080 A100 B600 0400 0100 0123 0000 A100 B600 0400 0C00 80A1 00B6 0004 0001"
        $"0001 2300 00A1 00B6 0004 0040 0080 A100 B600 0400 0100 0123 0000 A100 B600 0400"
        $"4000 80A1 00B6 0004 0001 0001 2300 00A1 00B6 0004 0040 0080 A100 B600 0400 0100"
        $"0123 0000 A100 B600 0400 4000 80A1 00B6 0004 0001 0001 2300 00A1 00B6 0004 0040"
        $"0080 A100 B600 0400 0100 0123 0000 A100 B600 0400 4000 80A1 00B6 0004 0001 0001"
        $"2300 00A1 00B6 0004 0040 0080 A100 B600 0400 0100 0123 0000 A100 B600 0400 4000"
        $"80A1 00B6 0004 0001 0001 2300 00A1 00B6 0004 0040 0080 A100 B600 0400 0100 0123"
        $"0000 A100 B600 0400 4000 80A1 00B6 0004 0001 0001 2300 00A1 00B6 0004 0040 0080"
        $"A100 B600 0400 0100 0123 0000 A100 B600 0400 4000 80A1 00B6 0004 0001 0001 2300"
        $"00A1 00B6 0004 0040 0080 A100 B600 0400 0100 0123 0000 A100 B600 0400 4000 80A1"
        $"00B6 0004 0001 0001 2300 00A1 00B6 0004 0040 0080 A100 B600 0400 0100 0123 0000"
        $"A100 B600 0400 4000 80A1 00B6 0004 0001 0001 2300 00A1 00B6 0004 0040 0080 A100"
        $"B600 0400 0100 0123 0000 A100 B600 0400 4000 80A1 00B6 0004 0001 0001 2300 00A1"
        $"00B6 0004 0040 0080 A100 B600 0400 0100 0123 0000 A000 83FF"
};

resource 'CNTL' (128, "scroll thing") {
        {0, 155, 234, 170},
        0,
        visible,
        0,
        0,
        scrollBarProc,
        0,
        "scroll thing"
};

data 'CDEF' (133, "Popup menu") {
        $"600E 0000 4344 4546 0085 0000 0000 0000"            /* `...CDEF.Ö...... */
        $"41FA FFEE 21C8 09CE 6000 08AC 48E7 C0C0"            /* A...!»ΔŒ`..¨H.¿¿ */
        $"322F 0014 206F 0016 2248 7000 22C0 22C0"            /* 2/.. o.."Hp."¿"¿ */
        $"22C0 22C0 22C0 22C0 22C0 22C0 226F 001A"            /* "¿"¿"¿"¿"¿"¿"o.. */
        $"5341 671E 701F 9041 E249 41F0 0000 30D9"            /* SAg.p.êA.IA...0. */
        $"51C9 FFFC 4CDF 0303 2F57 000A 4FEF 000A"            /* Q....L.../W..O... */
        $"4E75 1159 001F 60EC 2F0A 226F 0008 246F"            /* Nu.Y..`./."o..$o */
        $"000C 302F 0010 3400 121A B202 6402 1401"            /* ..0/..4...≤.d... */
        $"12C2 6002 12DA 51CA FFFC B001 245F 205F"            /* .¬`...Q ..∞.$_ _ */
        $"4FEF 000A 4ED0 225F 201F A04C 2E80 7000"            /* O...N-"_ .†L.Äp. */
        $"2F09 31C0 0220 4E75 7000 60F6 225F 201F"            /* /Δ1¿. Nup.`."_ . */
        $"A122 2E88 4EFA FFEA 225F 205F A023 4EFA"            /* °".àN..."_ _†#N. */
        $"FFE0 225F 205F A029 4EFA FFD6 225F 205F"            /* .."_ _†)N..+"_ _ */
        $"A02A 4EFA FFCC 4E56 FFC6 206E 000C 43EE"            /* †*N..ÃNV.Δ n..C. */
        $"FFE8 22D8 22D8 1D7C 0055 FFCE 1D7C 00AA"             /* .."ÿ"ÿ.|.U.Œ.|.™ */
        $"FFCF 1D7C 0055 FFD0 1D7C 00AA FFD1 1D7C"             /* .œ.|.U.–.|.™.—.| */
        $"0055 FFD2 1D7C 00AA FFD3 1D7C 0055 FFD4"             /* .U.".|.™.".|.U.' */
        $"1D7C 00AA FFD5 486E FFD6 A898 A89E 486E"              /* .|.™.'Hn.+®ò®ûHn */
        $"FFCE A89D 3F3C 000B A89C 486E FFE8 A8A2"              /* .Œ®ù?<..®úHn..®¢ */
        $"486E FFD6 A899 4E5E 205F 504F 4ED0 4E56"             /* Hn.+®ôN^ _PON-NV */
        $"FDE8 48E7 0308 7E01 4246 422E FDEC 594F"            /* ..H...~.BFB...YO */
        $"3F2E 0010 A9BF 285F 200C 6748 2F0C 3F07"             /* ?...©ø(_ .gH/.?. */
        $"486E FEEC A946 554F 486E FEEC A88C BC5F"              /* Hn..©FUOHn..®åº_ */
        $"6C1A 3F3C 00FF 486E FEEC 486E FDEC 4EBA"            /* l.?<..Hn..Hn..N∫ */
        $"FEE8 554F 486E FEEC A88C 3C1F 5247 0C47"             /* ..UOHn..®å<.RG.G */
        $"0064 5EC0 7200 122E FEEC 4A41 57C1 8001"            /* .d^¿r.....JAW¡Ä. */
        $"67BA 6016 41FA 0038 43EE FDEC 22D8 22D8"            /* g∫`.A..8C..."ÿ"ÿ */
        $"554F 486E FEEC A88C 3C1F 3D46 0012 206E"            /* UOHn..®å<.=F.. n */
        $"000C 3F3C 00FF 486E FDEC 2F08 4EBA FE9A"            /* ..?<..Hn../.N∫.ö */
        $"4CDF 10C0 4E5E 205F 4FEF 000A 4ED0 043F"            /* L..¿N^ _O...N-.? */
        $"3F3F 3F08 4E56 FEEC 2F0C 594F 3F2E 000E"            /* ???.NV../.YO?... */
        $"A9BF 285F 200C 6742 2F0C 3F2E 000C 486E"            /* ©ø(_ .gB/.?...Hn */
        $"FEEC A946 7000 102E FEEC 4A40 6F08 486E"            /* ..©Fp.....J@o.Hn */
        $"FEEC A884 602A 2F0C 3F3C 0001 486E FEEC"            /* ..®Ñ`*..?<..Hn.. */
        $"A946 206E 0008 2068 0010 2050 317C 0001"            /* ©F n.. h.. P1|.. */
```

```
$"0012 486E FEEC A884 6006 487A 000E A884"          /* ..Hn..®Ñ`.Hz..®Ñ */
$"285F 4E5E 205F 504F 4ED0 033F 3F3F 4E56"          /* (_N^ _PON-.???NV */
$"FB46 48E7 1F38 286E 0008 3D7C FFE0 FB78"          /* .FH..8(n..=|...x */
$"3D7C 7FC0 FB7A 3D7C 3F80 FB7C 3D7C 1F00"          /* =|.¿.z=|?Ä.|=|.. */
$"FB7E 3D7C 0E00 FB80 3D7C 0400 FB82 2F2E"          /* .~=|...Ä=|...Ç/. */
$"0010 4EBA FE3E 206E 0010 2050 7000 1028"          /* ..N∫.> n.. Pp..( */
$"0010 4A40 6700 03B4 206E 0010 2050 41E8"          /* ..J@g..¥ n.. PA. */
$"0008 43EE FFE8 22D8 22D8 486E FFBE A898"          /* ..C..."ÿ"ÿHn.æ®ò */
$"A89E 486E FFB6 A874 206E FFB6 3628 0044"          /* ®ûHn.∂®t n.∂6(.D */
$"206E FFB6 3828 004A 206E FFB6 3A28 0048"          /*  n.∂8(.J n.∂:(.H */
$"206E FFB6 4868 0046 486E FB48 3F3C 0001"          /*  n.∂Hh.FHn.H?<.. */
$"4EBA FD3A 1D6E FB67 FFAF 4267 A887 3F3C"          /* N∫.:.n.g.ØBg®á?< */
$"000C A88A 3F3C 0001 A889 422E FB66 7000"          /* ..®ä?<..®âB..fp. */
$"102E FB66 3F00 A888 486C FFE4 A88B 594F"          /* ...f?.®àHl..®ãYO */
$"A8D8 2E1F 2F07 A87A 594F A8D8 245F 2F0A"          /* ®ÿ../.®zYO®ÿ$_/. */
$"486E FFE8 A8DF 2F07 2F0A 2F0A A8E4 2F0A"          /* Hn..®./././.®./. */
$"A879 486E FFE8 A8A3 206E 0010 2050 4AA8"          /* ®yHn..®£ n.. PJ® */
$"001C 6600 0162 594F 7012 2F00 4EBA FD5E"          /* ..f..bYOp./.N∫.^ */
$"295F FFEC 206E 0010 2050 216C FFEC 001C"          /* )_.. n.. P!l.... */
$"2F2C FFEC 4EBA FD5C 206C FFEC 2650 4253"          /* /,..N∫.\ l..&PBS */
$"426B 0002 426B 0004 426B 0006 426B 0008"          /* Bk..Bk..Bk..Bk.. */
$"41EE FFE8 43EB 000A 22D8 22D8 594F 2F3C"          /* A...C..."ÿ"ÿYO/< */
$"4D45 4E55 206E 0010 2050 3F28 0016 A9A0"          /* MENU n.. P?(..©† */
$"2C1F 4A86 6712 2F06 486E FCA6 486E FCA2"          /* ,.JÜg./.Hn.¶Hn.¢ */
$"486E FCAC A9A8 600C 41FA 02A6 43EE FCAC"          /* Hn.¨©®`.A..¶C..¨ */
$"22D8 22D8 206E 0010 2050 3F3C 00FF 486E"          /* "ÿ"ÿ n.. P?<..Hn */
$"FCAC 4868 0028 4EBA FC90 554F 486E FCAC"          /* .¨Hh.(N∫.êUOHn.¨ */
$"A88C 206C FFEC 2050 309F 206C FFEC 2050"          /* ®å l.. P0ü l.. P */
$"4A50 6F0C 206C FFEC 2050 317C 0005 0002"          /* JPo. l.. P1|.... */
$"302C FFE4 5240 5240 206C FFEC 2050 3140"          /* 0,..R@R@ l.. P1@ */
$"0004 554F 206E 0010 2050 3F28 0016 486E"          /* ..UO n.. P?(..Hn */
$"FBA2 2F0E 4EBA FD18 3D5F FFE0 486E FFD0"          /* .¢/.N∫..=_..Hn.- */
$"4267 4267 302E FFE0 0640 000D 5A40 3F00"          /* BgBg0....@.¬Z@?. */
$"302C FFE4 D06C FFE6 5440 3F00 A8A7 486E"          /* 0,..-l..T@?.®ßHn */
$"FFD0 206C FFEC 2050 226C FFEC 2251 3010"          /* .- l.. P"l.."Q0. */
$"D069 0002 5640 3F00 3F3C 0001 A8A8 206C"          /* -i..V@?.?<..®® l */
$"FFEC 2050 43EE FFD0 41E8 000A 20D9 20D9"          /* .. PC..-A... . . */
$"302E FFD2 0640 000D 206C FFEC 2050 3140"          /* 0..".@.¬ l.. P1@ */
$"0006 302E FFD0 D06C FFE4 5240 206C FFEC"          /* ..0..--l..R@ l.. */
$"2050 3140 0008 206E 0010 2050 2968 001C"          /*  P1@.. n.. P)h.. */
$"FFEC 2F2C FFEC 4EBA FC0A 206C FFEC 2050"          /* ../,..N∫.. l.. P */
$"302E FFEA D068 0002 3F00 206C FFEC 2050"          /* 0...-h..?. l.. P */
$"302E FFE8 D068 0004 3F00 A893 206E 0010"          /* 0...-h..?.®ì n.. */
$"2050 7000 1028 0028 4A40 6F0C 206E 0010"          /*  Pp..(.(J@o. n.. */
$"2050 4868 0028 A884 206C FFEC 2050 41E8"          /*  PHh.(®Ñ l.. PA. */
$"000A 43EE FFD0 22D8 22D8 486E FFD0 3F2E"          /* ..C..-"ÿ"ÿHn.-?. */
$"FFEA 3F2E FFE8 A8A8 206C FFEC 2050 302E"          /* ..?...®® l.. P0. */
$"FFEA D068 0006 3F00 206C FFEC 2050 302E"          /* ..-h..?. l.. P0. */
$"FFE8 D068 0008 3F00 A893 206E 0010 2050"          /* ..-h..?.®ì n.. P */
$"3F28 0016 206E 0010 2050 3F28 0012 2F0E"          /* ?(.. n.. P?(../. */
$"4EBA FC92 486E FFD0 A8A1 302E FFD2 5240"          /* N∫.íHn.-®°0.."R@ */
$"3F00 3F2E FFD4 A893 3F2E FFD6 3F2E FFD4"          /* ?.?..'®ì?..÷?..' */
$"A891 3F2E FFD6 302E FFD0 5240 3F00 A891"          /* ®ë?..÷0..-R@?.®ë */
$"486E FB8C 4267 4267 3F3C 0010 3F3C 0006"          /* Hn.åBgBg?<..?<.. */
$"A8A7 41EE FB78 2D48 FB94 3D7C 0002 FB98"          /* ®ßA..x-H.î=|...ò */
$"41EE FB8C 43EE FB9A 22D8 22D8 41EE FB8C"          /* A..åC..ö"ÿ"ÿA..å */
$"43EE FB84 22D8 22D8 486E FB84 302E FFD6"          /* C..Ñ"ÿ"ÿHn.Ñ0..÷ */
$"0640 FFEF 3F00 302E FFD0 5C40 3F00 A8A8"          /* .@..?.0..-\@?.®® */
$"486E FFBA A874 486E FB94 206E FFBA 4868"          /* Hn.∫®tHn.î n.∫Hh */
$"0002 486E FB8C 486E FB84 4267 42A7 A8EC"          /* ..Hn.åHn.ÑBgBß®. */
$"206E 0010 2050 7000 1028 0011 0C40 00FF"          /*  n.. Pp..(...@.. */
$"660A 486E FFE8 2F0E 4EBA FACC 2F07 A879"          /* f.Hn../.N∫.À/.®y */
$"2F07 A8D9 2F0A A8D9 3F03 A887 3F04 A88A"          /* /.®./.®.?.®á?.®ä */
$"3F05 A889 486E FFAF 486E FB48 3F3C 0001"          /* ?.®âHn.ØHn.H?<.. */
```

```
        $"4EBA F9EA 1D6E FB67 FB46 7000 102E FB46"            /* N∫...n.g.Fp....F */
        $"3F00 A888 486E FFBE A899 2F2E 0010 4EBA"            /* ?.®àHn.æ®ô/...N∫ */
        $"FA7C 4CDF 1CF8 4E5E 205F 4FEF 000E 4ED0"            /* .|L...N^ _O...N- */
        $"043F 3F3F 3F08 4E56 FFE8 2F0C 286E 0010"            /* .????.NV../.(n.. */
        $"42AE 0016 2F0C 4EBA FA4A 2054 7000 1028"            /* BÆ../.N∫.J Tp..( */
        $"0011 0C40 00FF 6762 2054 7000 1028 0011"            /* ...@..gb Tp..(.. */
        $"0C40 00FE 674C 2054 41E8 0008 43EE FFE8"            /* .@..gL TA...C... */
        $"22D8 22D8 302E FFEE 5B40 3D40 FFEA 302E"            /* "ÿ"ÿ0...[@=@..0. */
        $"FFEC 5B40 3D40 FFE8 554F 2F2E 000C 2054"            /* ..[@=@..UO/... T */
        $"4868 0008 A8AD 554F 2F2E 000C 486E FFE8"            /* Hh..®≠UO/...Hn.. */
        $"A8AD 101F 5300 C01F 6710 700A 2D40 0016"            /* ®≠..S.¿.g.p.-@.. */
        $"6008 2D7C 0000 00FE 0016 2F0C 4EBA F9DE"            /* `.-|....../.N∫.. */
        $"285F 4E5E 205F 4FEF 000E 4ED0 4E56 FF78"            /* (_N^ _O...N-NV.x */
        $"48E7 1F38 286E 000C 266E 0008 2054 3A28"            /* H..8(n..&n.. T:( */
        $"0016 594F 3F05 A9BF 245F 200A 6700 019C"            /* ..YO?.©ø$_ .g..ú */
        $"486E FFC4 A898 A89E 486E FFC0 A874 206E"            /* Hn.ƒ®ò®ûHn.¿®t n */
        $"FFC0 3D68 0044 FFBE 206E FFC0 3628 004A"            /* .¿=h.D.æ n.¿6(.J */
        $"206E FFC0 3828 0048 206E FFC0 4868 0046"            /*  n.¿8(.H n.¿Hh.F */
        $"486E FF7A 3F3C 0001 4EBA F8C2 1D6E FF99"            /* Hn.z?<..N∫.¬.n.ô */
        $"FFB9 4267 A887 3F3C 000C A88A 3F3C 0001"            /* .πBg®á?<..®ä?<.. */
        $"A889 422E FF98 7000 102E FF98 3F00 A888"            /* ®âB..òp....ò?.®à */
        $"486B FFE4 A88B 2F0A 3F3C FFFF A935 2054"            /* Hk..®ã/.?<..©5 T */
        $"2768 001C FFEC 2F2B FFEC 4EBA F926 2054"            /* 'h..../+..N∫.& T */
        $"41E8 0008 43EE FFD6 22D8 22D8 206B FFEC"            /* A...C..+"ÿ"ÿ k.. */
        $"2050 41E8 000A 43EE FFDE 22D8 22D8 486E"            /*  PA...C..."ÿ"ÿHn */
        $"FFDE 3F2E FFD8 3F2E FFD6 A8A8 3D6E FFE0"            /* ..?..ÿ?..+®®=n.. */
        $"FFE4 3D6E FFD8 FFE0 2054 7000 1028 0028"            /* ..=n.ÿ.. Tp..(.( */
        $"4A40 6F06 486E FFDE A8A4 3D6E FFD6 FFEC"            /* J@o.Hn..®§=n.+.. */
        $"206B FFEC 2050 302E FFD8 D068 000C 3D40"            /*  k.. P0..ÿ-h..=@ */
        $"FFEE 486E FFEC A870 2F0A 2054 3F28 0012"            /* ..Hn..®p/. T?(.. */
        $"1F3C 0001 A945 594F 2F0A 3F2E FFEC 3F2E"            /* .<..©EYO/.?...?. */
        $"FFEE 2054 3F28 0012 A80B 2E1F 4206 2054"            /* .. T?(..®...B. T */
        $"BE68 0012 56C0 4A47 5EC1 C001 6716 2F0A"            /* æh..V¿JG^¡¿.g./. */
        $"2054 3F28 0012 4227 A945 2054 3147 0012"            /*  T?(..B'©E T1G.. */
        $"7C01 600C 2F0A 2054 3F28 0012 4227 A945"            /* |.`./. T?(..B'©E */
        $"3F05 A936 2054 7000 1028 0028 4A40 6F06"            /* ?.©6 Tp..(.(J@o. */
        $"486E FFDE A8A4 4A06 670C 4267 2F0C 42A7"            /* Hn..®§J.g.Bg/.BB */
        $"2F0B 4EBA F9CA 3F2E FFBE A887 3F03 A88A"            /* /.N∫. ?..æ®á?.®ä */
        $"3F04 A889 486E FFB9 486E FF7A 3F3C 0001"            /* ?.®âHn.πHn.z?<.. */
        $"4EBA F77A 1D6E FF99 FF78 7000 102E FF78"            /* N∫.z.n.ô.xp....x */
        $"3F00 A888 486E FFC4 A899 4CDF 1CF8 4E5E"            /* ?.®àHn.ƒ®ôL...N^ */
        $"205F 504F 4ED0 4E56 FFE4 2F0C 286E 000E"            /*  _PON-NV../.(n.. */
        $"42AE 0014 200C 6700 00AA 302E 000C 6000"            /* BÆ.. .g..™0...`. */
        $"008A 3F2E 0012 2F0C 2F2E 0008 2F0E 4EBA"            /* .ä?.../.../.N∫ */
        $"F95E 6000 008E 594F 3F2E 0012 2F0C 2F2E"            /* .^`..éYO?.../ ./. */
        $"0008 2F0E 4EBA FD60 2D5F 0014 6000 0074"            /* ../.N∫.`-_..`..t */
        $"2F0C 4EBA F7AE 2F2E 0008 2054 4868 0008"            /* /.N∫.Æ/... THh.. */
        $"A8DF 2F0C 4EBA F7A6 6058 2F0C 4EBA F794"            /* ®./.N∫.¶`X/.N∫.î */
        $'2054 70FF 2140 0020 2F0C 4EBA F790 6042"            /*  Tp.!@. /.N∫.ê`B */
        $"2054 4AA8 001C 673A 2054 2D68 001C FFEC"            /*  TJ®..g: T-h.... */
        $"2F2E FFEC 4EBA F762 2054 42A8 001C 6022"            /* /...N∫.b TB®..`" */
        $"2F0C 2F0E 4EBA FD96 6018 6700 FF76 5340"            /* /./.N∫.ñ`.g..vS@ */
        $"6784 5340 679A 5340 67B0 5340 67C2 5940"            /* gÑS@göS@g∞S@g¬Y@ */
        $"67DE 285F 4E5E 205F 4FEF 000C 4ED0"                 /* g.(_N^ _O...N- */
};

resource 'CNTL' (133, "Popup menu") {
        {0, 0, 31, 31},
        1,
        invisible,
        100,
        1,
        2128,
        133,
        "Control for Dialog Simulation"
        /****** Extra bytes follow... ******/
        /* $"0000"                                        /* .. */
};

resource 'SIZE' (-1) {
        dontSaveScreen,
        acceptSuspendResumeEvents,
        enableOptionSwitch,
        canBackground,
        multiFinderAware,
        backgroundAndForeground,
        dontGetFrontClicks,
        ignoreChildDiedEvents,
        is32BitCompatible,
        reserved,
        reserved,
        reserved,
        reserved,
        reserved,
        reserved,
        reserved,
        1048576,
        1048576
};
```

10.5 NODDS DATA RETRIEVAL CODE

This section contains the FORTRAN source code and the MS-DOS batch files which CRC used on an IBM-PC compatible computer to use the Navy Oceanographic Data Distribution System (NODDS) to obtain atmospheric data for use with the BDPS program. Though this code is not a formal portion of the BDPS program, we have included it as an example of a means of providing wind data to BDPS from a source other than GRAM.

10.5.1 NODDS Batch Files

The two batch files (.BAT) included here install the support files for collecting and formatting the data and then configure the downloaded data files to support the production of wind table data for use with BDPS.

```
copy wind.bat c:\nodds
copy read.exe c:\nodds\fields
copy getdir.exe c:\nodds\fields
c:
cd c:\nodds
```

```
cd fields
\nodds\makebat2
getdir
call d
\nodds\decodfld
rem
copy c00*.??g alt1.tmp
copy c20*.??g east1.tmp
copy c21*.??g north1.tmp
```

```
rem
copy r00*.??g alt2.tmp
copy r20*.??g east2.tmp
copy r21*.??g north2.tmp
rem
copy d00*.??g alt3.tmp
copy d20*.??g east3.tmp
copy d21*.??g north3.tmp
rem
copy e00*.??g alt4.tmp
copy e20*.??g east4.tmp
copy e21*.??g north4.tmp
rem
copy f00*.??g alt5.tmp
copy f20*.??g east5.tmp
copy f21*.??g north5.tmp
rem
copy g00*.??g alt6.tmp
copy g20*.??g east6.tmp
copy g21*.??g north6.tmp
rem
copy h00*.??g alt7.tmp
copy h20*.??g east7.tmp
copy h21*.??g north7.tmp
rem
copy t00*.??g alt8.tmp
copy t20*.??g east8.tmp
copy t21*.??g north8.tmp
rem
copy i00*.??g alt9.tmp
copy i20*.??g east9.tmp
copy i21*.??g north9.tmp
read
copy wind1.dat b:east.dat
copy wind2.dat b:north.dat
del *.??g
del *.tmp
del d.bat
del wind1.dat
del wind2.dat
del runbatch.bat
cd ..
```

10.5.2 NODDS FORTRAN Files

These files were written to process the downloaded NODDS data into a format suitable for transferring to the Macintosh for use with BDPS.

```
C***************************************************************************C
C                                                                           C
C                                                                           C
C     THIS PROGRAM EXTRACTS THE FIRST TWO LINES OF THE RUNBATCH.BAT         C
C     FILE TO GET TO THE MAPTXT.DAT FILE.                                   C
C                                                                           C
C                                                                           C
C***************************************************************************C
C
C
      CHARACTER DIR*80
      OPEN(UNIT=1,STATUS='OLD',FILE='RUNBATCH.BAT')
      OPEN(UNIT=2,STATUS='NEW',FILE='D.BAT')
      DO I=1,2
      READ(1,10)DIR
      WRITE(2,10)DIR
      END DO
 10   FORMAT(A80)
      CLOSE(1)
      CLOSE(2)
      STOP
      END
```

```
C**********************************************************************C
C                                                                      C
C                                                                      C
C      THIS PROGRAM IS USED TO READ THE TEMPORARY FILES CREATED BY     C
C                                                                      C
C      WIND.BAT.  THE PROGRAM READS THESE FILES AND PUTS THE DATA IN   C
C                                                                      C
C      A FORM THAT THE SUBROUTINE NOGAPS CAN READ.                     C
C                                                                      C
C**********************************************************************C
C                                                                      C
C                                                                      C
       DIMENSION A(1499),E(1499),N(1499),ALT(1499),EAST(1499)
       DIMENSION LONT(1499),LAT(1499),NORTH(1499)
       INTEGER I,M,L10,L20,L30,L40,MG,NG,J1
       REAL     ALT,LAT,LONT,EAST,NORTH,LAT0,LATB,LONT0,LONTR,SLAT,SLONT
       REAL     NUMBALT,MGRD,NGRD,MAT,A,E,N
       CHARACTER DIR1,DIR2,DIR3,DIR4
       CHARACTER*6 L11,L22,L33,L44
C**********************************************************************C
C                                                                      C
C                                                                      C
C      OPEN THE FILE THAT DEFINES THE AREA OF INTEREST
C
C
       OPEN(UNIT=78,STATUS='OLD',FILE='MAPTEMP.DAT')
C
C
C      READ THE RANGE OF LATITUDES,LONGITUDES, AND WHAT QUADRANT(S)
C      THEY EXIST.
C
C
      READ(78,6)
      READ(78,4000)L11,L22
      READ(78,4000)L33,L44
 4000 FORMAT(19X,A6,/,19X,A6)
      REWIND(78)
      READ(78,6)
      POS1=INDEX(L11,'.')
      POS2=INDEX(L22,'.')
      POS3=INDEX(L33,'.')
      POS4=INDEX(L44,'.')
      IF (POS1.LE.0)THEN
         READ(78,1)L10,DIR1
         LAT0=FLOAT(L10)
      ELSE IF (POS1.EQ.2)THEN
         READ(78,2)LAT0,DIR1
      ELSE IF (POS1.EQ.3)THEN
         READ(78,3)LAT0,DIR1
      END IF
      IF (POS2.LE.0)THEN
         READ(78,1)L20,DIR2
         LATB=FLOAT(L20)
      ELSE IF (POS2.EQ.2)THEN
```

```
         READ(78,2)LATB,DIR2
      ELSE IF (POS2.EQ.3)THEN
         READ(78,3)LATB,DIR2
      END IF
      IF (POS3.LE.0)THEN
         READ(78,1)L30,DIR3
         LONTR=FLOAT(L30)
      ELSE IF (POS3.EQ.2)THEN
         READ(78,2)LONTR,DIR3
      ELSE IF (POS3.EQ.3)THEN
         READ(78,3)LONTR,DIR3
      ELSE IF (POS3.EQ.4)THEN
         READ(78,4)LONTR,DIR3
      END IF
      IF (POS4.LE.0)THEN
         READ(78,1)L40,DIR4
         LONTO=FLOAT(L40)
      ELSE IF (POS4.EQ.2)THEN
         READ(78,2)LONTO,DIR4
      ELSE IF (POS4.EQ.3)THEN
         READ(78,3)LONTO,DIR4
      ELSE IF (POS4.EQ.4)THEN
         READ(78,4)LONTO,DIR4
      END IF
      DO I=1,2
      READ(78,6)
      END DO
      READ(78,7)MG,NG
      MGRD=FLOAT(MG)
      NGRD=FLOAT(NG)
1     FORMAT(19X,I2,A1)
2     FORMAT(19X,F3.1,A1)
3     FORMAT(19X,F4.1,A1)
4     FORMAT(19X,F5.1,A1)
6     FORMAT(80X)
7     FORMAT(9X,I2,8X,I2)
      MAT=MGRD*NGRD
C
C
C     CALCULATE GRID SCALES
C
C
      IF (DIR1.EQ.'S')THEN
         LATO=-LATO
      END IF
      IF (DIR2.EQ.'S')THEN
         LATB=-LATB
      END IF
      IF (DIR3.EQ.'W')THEN
         LONTR=-LONTR
      END IF
      IF (DIR4.EQ.'W')THEN
         LONTO=-LONTO
      END IF
C
      SLAT=-(LATO-LATB)/(NGRD-1)
      IF ((DIR3.EQ.'W').AND.(DIR4.EQ.'E'))THEN
         SLONT=((180-LONTO)+(180+LONTR))/(MGRD-1)
      ELSE
         SLONT=-(LONTO-LONTR)/(MGRD-1)
      END IF
C
C
C*********************************************************************C
```

```
C                                                                    C
C                                                                    C
C     READ THE 1000MB WIND DATA                                      C
C                                                                    C
C                                                                    C
C********************************************************************C
C
C
      OPEN(UNIT=50,STATUS='OLD',FILE='ALT1.TMP')
      OPEN(UNIT=51,STATUS='OLD',FILE='EAST1.TMP')
      OPEN(UNIT=52,STATUS='OLD',FILE='NORTH1.TMP')
C
      DO I=1,2
      READ(50,10)
      READ(51,10)
      READ(52,10)
      END DO
C
      DO I=1,MAT,13
      READ(50,15)A(I),A(I+1),A(I+2),A(I+3),A(I+4),A(I+5),A(I+6),A(I+7),
     + A(I+8),A(I+9),A(I+10),A(I+11),A(I+12)
      READ(51,15)E(I),E(I+1),E(I+2),E(I+3),E(I+4),E(I+5),E(I+6),E(I+7),
     + E(I+8),E(I+9),E(I+10),E(I+11),E(I+12)
      READ(52,15)N(I),N(I+1),N(I+2),N(I+3),N(I+4),N(I+5),N(I+6),N(I+7),
     + N(I+8),N(I+9),N(I+10),N(I+11),N(I+12)
      END DO
C
      CLOSE(50)
      CLOSE(51)
      CLOSE(52)
C
C
C********************************************************************C
C                                                                    C
C                                                                    C
C     READ THE 925MB WIND DATA                                       C
C                                                                    C
C                                                                    C
C********************************************************************C
C
C
      OPEN(UNIT=53,STATUS='OLD',FILE='ALT2.TMP')
      OPEN(UNIT=54,STATUS='OLD',FILE='EAST2.TMP')
      OPEN(UNIT=55,STATUS='OLD',FILE='NORTH2.TMP')
C
      DO I=1,2
      READ(53,10)
      READ(54,10)
      READ(55,10)
      END DO
C
      DO I=(MAT+1),(2*MAT),13
      READ(53,15)A(I),A(I+1),A(I+2),A(I+3),A(I+4),A(I+5),A(I+6),A(I+7),
     + A(I+8),A(I+9),A(I+10),A(I+11),A(I+12)
      READ(54,15)E(I),E(I+1),E(I+2),E(I+3),E(I+4),E(I+5),E(I+6),E(I+7),
     + E(I+8),E(I+9),E(I+10),E(I+11),E(I+12)
      READ(55,15)N(I),N(I+1),N(I+2),N(I+3),N(I+4),N(I+5),N(I+6),N(I+7),
     + N(I+8),N(I+9),N(I+10),N(I+11),N(I+12)
      END DO
C
      CLOSE(53)
      CLOSE(54)
      CLOSE(55)
C
```

```
C
C**********************************************************************C
C                                                                      C
C                                                                      C
C     READ THE 850MB WIND DATA                                         C
C                                                                      C
C                                                                      C
C**********************************************************************C
C
C
      OPEN(UNIT=56,STATUS='OLD',FILE='ALT3.TMP')
      OPEN(UNIT=57,STATUS='OLD',FILE='EAST3.TMP')
      OPEN(UNIT=58,STATUS='OLD',FILE='NORTH3.TMP')
C
      DO I=1,2
      READ(56,10)
      READ(57,10)
      READ(58,10)
      END DO
C
      DO I=(2*MAT+1),(3*MAT),13
      READ(56,15)A(I),A(I+1),A(I+2),A(I+3),A(I+4),A(I+5),A(I+6),A(I+7),
     + A(I+8),A(I+9),A(I+10),A(I+11),A(I+12)
      READ(57,15)E(I),E(I+1),E(I+2),E(I+3),E(I+4),E(I+5),E(I+6),E(I+7),
     + E(I+8),E(I+9),E(I+10),E(I+11),E(I+12)
      READ(58,15)N(I),N(I+1),N(I+2),N(I+3),N(I+4),N(I+5),N(I+6),N(I+7),
     + N(I+8),N(I+9),N(I+10),N(I+11),N(I+12)
      END DO
C
      CLOSE(56)
      CLOSE(57)
      CLOSE(58)
C
C
C**********************************************************************C
C                                                                      C
C                                                                      C
C     READ THE 700MB WIND DATA                                         C
C                                                                      C
C                                                                      C
C**********************************************************************C
C
C
      OPEN(UNIT=59,STATUS='OLD',FILE='ALT4.TMP')
      OPEN(UNIT=60,STATUS='OLD',FILE='EAST4.TMP')
      OPEN(UNIT=61,STATUS='OLD',FILE='NORTH4.TMP')
C
      DO I=1,2
      READ(59,10)
      READ(60,10)
      READ(61,10)
      END DO
C
      DO I=3*MAT+1,4*MAT,13
      READ(59,15)A(I),A(I+1),A(I+2),A(I+3),A(I+4),A(I+5),A(I+6),A(I+7),
     + A(I+8),A(I+9),A(I+10),A(I+11),A(I+12)
      READ(60,15)E(I),E(I+1),E(I+2),E(I+3),E(I+4),E(I+5),E(I+6),E(I+7),
     + E(I+8),E(I+9),E(I+10),E(I+11),E(I+12)
      READ(61,15)N(I),N(I+1),N(I+2),N(I+3),N(I+4),N(I+5),N(I+6),N(I+7),
     + N(I+8),N(I+9),N(I+10),N(I+11),N(I+12)
      END DO
C
      CLOSE(59)
      CLOSE(60)
      CLOSE(61)
C
C
C**********************************************************************C
C                                                                      C
C                                                                      C
C     READ THE 500MB WIND DATA                                         C
C                                                                      C
C                                                                      C
C**********************************************************************C
C
C
      OPEN(UNIT=62,STATUS='OLD',FILE='ALT5.TMP')
      OPEN(UNIT=63,STATUS='OLD',FILE='EAST5.TMP')
      OPEN(UNIT=64,STATUS='OLD',FILE='NORTH5.TMP')
C
      DO I=1,2
      READ(62,10)
      READ(63,10)
      READ(64,10)
      END DO
C
      DO I=4*MAT+1,5*MAT,13
      READ(62,15)A(I),A(I+1),A(I+2),A(I+3),A(I+4),A(I+5),A(I+6),A(I+7),
     + A(I+8),A(I+9),A(I+10),A(I+11),A(I+12)
      READ(63,15)E(I),E(I+1),E(I+2),E(I+3),E(I+4),E(I+5),E(I+6),E(I+7),
     + E(I+8),E(I+9),E(I+10),E(I+11),E(I+12)
      READ(64,15)N(I),N(I+1),N(I+2),N(I+3),N(I+4),N(I+5),N(I+6),N(I+7),
     + N(I+8),N(I+9),N(I+10),N(I+11),N(I+12)
      END DO
C
      CLOSE(62)
      CLOSE(63)
      CLOSE(64)
C
C
C**********************************************************************C
C                                                                      C
C                                                                      C
C     READ THE 400MB WIND DATA                                         C
C                                                                      C
C                                                                      C
C**********************************************************************C
C
C
      OPEN(UNIT=65,STATUS='OLD',FILE='ALT6.TMP')
      OPEN(UNIT=66,STATUS='OLD',FILE='EAST6.TMP')
      OPEN(UNIT=67,STATUS='OLD',FILE='NORTH6.TMP')
C
      DO I=1,2
      READ(65,10)
      READ(66,10)
      READ(67,10)
      END DO
C
      DO I=5*MAT+1,6*MAT,13
      READ(65,15)A(I),A(I+1),A(I+2),A(I+3),A(I+4),A(I+5),A(I+6),A(I+7),
     + A(I+8),A(I+9),A(I+10),A(I+11),A(I+12)
      READ(66,15)E(I),E(I+1),E(I+2),E(I+3),E(I+4),E(I+5),E(I+6),E(I+7),
     + E(I+8),E(I+9),E(I+10),E(I+11),E(I+12)
      READ(67,15)N(I),N(I+1),N(I+2),N(I+3),N(I+4),N(I+5),N(I+6),N(I+7),
     + N(I+8),N(I+9),N(I+10),N(I+11),N(I+12)
      END DO
C
```

```
      CLOSE(65)
      CLOSE(66)
      CLOSE(67)
C
C
C**********************************************************************C
C                                                                      C
C                                                                      C
C     READ THE 300MB WIND DATA                                         C
C                                                                      C
C                                                                      C
C**********************************************************************C
C
C
      OPEN(UNIT=68,STATUS='OLD',FILE='ALT7.TMP')
      OPEN(UNIT=69,STATUS='OLD',FILE='EAST7.TMP')
      OPEN(UNIT=70,STATUS='OLD',FILE='NORTH7.TMP')
C
      DO I=1,2
      READ(68,10)
      READ(69,10)
      READ(70,10)
      END DO
C
      DO I=6*MAT+1,7*MAT,13
      READ(68,15)A(I),A(I+1),A(I+2),A(I+3),A(I+4),A(I+5),A(I+6),A(I+7),
     + A(I+8),A(I+9),A(I+10),A(I+11),A(I+12)
      READ(69,15)E(I),E(I+1),E(I+2),E(I+3),E(I+4),E(I+5),E(I+6),E(I+7),
     + E(I+8),E(I+9),E(I+10),E(I+11),E(I+12)
      READ(70,15)N(I),N(I+1),N(I+2),N(I+3),N(I+4),N(I+5),N(I+6),N(I+7),
     + N(I+8),N(I+9),N(I+10),N(I+11),N(I+12)
      END DO
C
      CLOSE(68)
      CLOSE(69)
      CLOSE(70)
C
C
C**********************************************************************C
C                                                                      C
C                                                                      C
C     READ THE 250MB WIND DATA                                         C
C                                                                      C
C                                                                      C
C**********************************************************************C
C
C
      OPEN(UNIT=71,STATUS='OLD',FILE='ALT8.TMP')
      OPEN(UNIT=72,STATUS='OLD',FILE='EAST8.TMP')
      OPEN(UNIT=73,STATUS='OLD',FILE='NORTH8.TMP')
C
      DO I=1,2
      READ(71,10)
      READ(72,10)
      READ(73,10)
      END DO
C
      DO I=7*MAT+1,8*MAT,13
      READ(71,15)A(I),A(I+1),A(I+2),A(I+3),A(I+4),A(I+5),
     + A(I+6),A(I+7),A(I+8),A(I+9),A(I+10),A(I+11),A(I+12)
      READ(72,15)E(I),E(I+1),E(I+2),E(I+3),E(I+4),E(I+5),E(I+6),E(I+7),
     + E(I+8),E(I+9),E(I+10),E(I+11),E(I+12)
      READ(73,15)N(I),N(I+1),N(I+2),N(I+3),N(I+4),N(I+5),N(I+6),N(I+7),
     + N(I+8),N(I+9),N(I+10),N(I+11),N(I+12)
```

```
      END DO
C
      CLOSE(71)
      CLOSE(72)
      CLOSE(73)
C
C
C**********************************************************************C
C                                                                      C
C                                                                      C
C     READ THE 200MB WIND DATA                                         C
C                                                                      C
C                                                                      C
C**********************************************************************C
C
C
      OPEN(UNIT=74,STATUS='OLD',FILE='ALT9.TMP')
      OPEN(UNIT=75,STATUS='OLD',FILE='EAST9.TMP')
      OPEN(UNIT=76,STATUS='OLD',FILE='NORTH9.TMP')
C
      DO I=1,2
      READ(74,10)
      READ(75,10)
      READ(76,10)
      END DO
C
      DO I=8*MAT+1,9*MAT,13
      READ(74,15)A(I),A(I+1),A(I+2),A(I+3),A(I+4),A(I+5),A(I+6),A(I+7),
     + A(I+8),A(I+9),A(I+10),A(I+11),A(I+12)
      READ(75,15)E(I),E(I+1),E(I+2),E(I+3),E(I+4),E(I+5),E(I+6),E(I+7),
     + E(I+8),E(I+9),E(I+10),E(I+11),E(I+12)
      READ(76,15)N(I),N(I+1),N(I+2),N(I+3),N(I+4),N(I+5),N(I+6),N(I+7),
     + N(I+8),N(I+9),N(I+10),N(I+11),N(I+12)
      END DO
C
      CLOSE(74)
      CLOSE(75)
      CLOSE(76)
C
 10   FORMAT(80X)
 15   FORMAT(13F6.0)
C
C**********************************************************************C
C                                                                      C
C                                                                      C
C     CONVERT INTEGER DATA TO REAL AND SCALE                           C
C                                                                      C
C                                                                      C
C**********************************************************************C
C
C
      SCALE=1
      DO J=1,9
         LONT((J-1)*MAT+1)=LONT0
         LAT((J-1)*MAT+1)=LAT0
      DO 20 I=(J-1)*MAT+1,J*MAT
         ALT(I)=A(I)/30.0
         EAST(I)=E(I)/19.44
         NORTH(I)=N(I)/19.44
C
         IF (I.EQ.SCALE*MGRD) THEN
            LONT(I+1)=LONT0
            LAT(I+1)=LAT(I)+SLAT
            SCALE=SCALE+1
```

```
          ELSE
             LONT(I+1)=LONT(I)+SLONT
             LAT(I+1)=LAT(I)
          END IF
          IF (LAT(I+1).GT.90.0)THEN
             LAT(I+1)=LAT(I+1)-180.0
          ELSE IF (LAT(I-1).LT.-90.0)THEN
             LAT(I+1)=LAT(I+1)+180.0
          END IF
          IF (LONT(I+1).GT.180.0)THEN
             LONT(I+1)=LONT(I+1)-360.0
          ELSE IF (LONT(I+1).LT.-180.0)THEN
             LONT(I+1)=LONT(I+1)+360.0
          END IF
20     CONTINUE
       END DO
C
C****************************************************************************C
C                                                                            C
C                                                                            C
C      SORT WIND DATA ON LATITUDE                                            C
C                                                                            C
C                                                                            C
C****************************************************************************C
C
C
       DO 200 I=1,9*MAT-1
         M=I+1
         DO 100 J=M,9*MAT
           IF (LAT(I) .GT. LAT(J)) THEN
              TEMP1=ALT(I)
              TEMP3=LAT(I)
              TEMP4=LONT(I)
              TEMP5=EAST(I)
              TEMP6=NORTH(I)
              ALT(I)=ALT(J)
              LAT(I)=LAT(J)
              LONT(I)=LONT(J)
              EAST(I)=EAST(J)
              NORTH(I)=NORTH(J)
              ALT(J)=TEMP1
              LAT(J)=TEMP3
              LONT(J)=TEMP4
              EAST(J)=TEMP5
              NORTH(J)=TEMP6
           END IF
100      CONTINUE
200    CONTINUE
C
C
C****************************************************************************C
C                                                                            C
C                                                                            C
C      SORT DATA ON LONGITUDE                                                C
C                                                                            C
C                                                                            C
C****************************************************************************C
C
C
       DO 310 J1=1,NGRD
       DO 210 I=(J1-1)*MGRD*9+1,J1*MGRD*9-1
         M=I+1
         DO 110 J=M,J1*MGRD*9
           IF (LONT(I) .GT. LONT(J)) THEN
```

```
              TEMP1=ALT(I)
              TEMP3=LAT(I)
              TEMP4=LONT(I)
              TEMP5=EAST(I)
              TEMP6=NORTH(I)
              ALT(I)=ALT(J)
              LAT(I)=LAT(J)
              LONT(I)=LONT(J)
              EAST(I)=EAST(J)
              NORTH(I)=NORTH(J)
              ALT(J)=TEMP1
              LAT(J)=TEMP3
              LONT(J)=TEMP4
              EAST(J)=TEMP5
              NORTH(J)=TEMP6
           END IF
110      CONTINUE
210    CONTINUE
310    CONTINUE
       DO 610 J1=1,9*MAT,9
       DO 510 I=(J1-1)+1,(J1-1)+8
         M=I+1
         DO 410 J=M,(J1-1)+9
           IF (ALT(I) .GT. ALT(J)) THEN
              TEMP1=ALT(I)
              TEMP3=LAT(I)
              TEMP4=LONT(I)
              TEMP5=EAST(I)
              TEMP6=NORTH(I)
              ALT(I)=ALT(J)
              LAT(I)=LAT(J)
              LONT(I)=LONT(J)
              EAST(I)=EAST(J)
              NORTH(I)=NORTH(J)
              ALT(J)=TEMP1
              LAT(J)=TEMP3
              LONT(J)=TEMP4
              EAST(J)=TEMP5
              NORTH(J)=TEMP6
           END IF
410      CONTINUE
510    CONTINUE
610    CONTINUE
C
C
C***************************************************************************C
C                                                                           C
C                                                                           C
C      WRITE THE DATA TO WIND1.DAT ( EAST WIND VECTOR )                     C
C      WRITE THE DATA TO WIND2.DAT ( NORTH WIND VECTOR )                    C
C                                                                           C
C                                                                           C
C***************************************************************************C
C
C
       OPEN(UNIT=77,STATUS='NEW',FILE='WIND1.DAT')
       OPEN(UNIT=79,STATUS='NEW',FILE='WIND2.DAT')
C
C
       NUMBALT=9.0
       WRITE(77,51)NUMBALT,MGRD,NGRD
       WRITE(77,*)
       WRITE(77,52) (ALT(I),I=1,9)
       WRITE(77,*)
```

```
        WRITE(77,52) (LONT(I),I=1,9*MGRD,9)
        WRITE(77,*)
        WRITE(77,52) (LAT(I),I=1,9*MAT,9*MGRD)
        WRITE(77,*)
C
        WRITE(79,51)NUMBALT,MGRD,NGRD
        WRITE(79,*)
        WRITE(79,52) (ALT(I),I=1,9)
        WRITE(79,*)
        WRITE(79,52) (LONT(I),I=1,9*MGRD,9)
        WRITE(79,*)
        WRITE(79,52) (LAT(I),I=1,9*MAT,9*MGRD)
        WRITE(79,*)
C
           DO J=1,9
           WRITE(77,50) (EAST(I),I=(J-1)*MAT+1,MAT*J)
           WRITE(79,50) (NORTH(I),I=(J-1)*MAT+1,MAT*J)
           END DO
C
50     FORMAT(F8.3,F8.3,F8.3,F8.3)
51     FORMAT(F8.3,F8.3,F8.3)
52     FORMAT(F9.3,F9.3,F9.3,F9.3)
C
C**********************************************************************C
C                                                                      C
C                                                                      C
C       CLOSE OUTPUT FILES                                             C
C                                                                      C
C                                                                      C
C**********************************************************************C
C
C
        CLOSE(77)
        CLOSE(79)
        STOP
        END
```

20.0 BDPS USER MANUAL

The purpose of this appendix to the final report is to describe the procedures with which a user may operate the Balloon Drift Pattern Simulation (BDPS).

20.1 BDPS CONCEPT OF OPERATIONS

CRC designea the BDPS tool to be easily operated for the analysis of a balloon's drift pattern as influenced primarily by atmospheric winds. BDPS was written for and tested on Macintosh computers with either 68020 or 68030 processors (and 68881 or 68882 floating point coprocessors). In general, a user supplies certain files according to the desired BDPS modeling options. The user starts the BDPS application and then takes a series of steps to generate, display, and save graphical representations of balloon drift patterns. The following sections describe the files involved and the steps to be taken to produce drift patterns with BDPS.

20.2 REQUIRED FILES FOR BDPS OPERATION

Besides the BDPS application program, several files are required for the successful operation of BDPS. Every BDPS run will require an ascent profile which contains data that represents a particular balloon configuration's vertical ascent rate as a function of time. Then, depending on the user's preference, the BDPS application will either require a set of climate files associated with the GRAM atmosphere model or a set of files for using the BDPS wind table option. These files are discussed in the following sections.

20.2.1 Ascent Profile

Every BDPS run will require an ascent profile which contains data that represents a particular balloon configuration's vertical ascent rate as a function of time. The file should be named "Ascent Profile" and should be located in the same Macintosh folder as the BDPS application. The file may be created with any text editor or word processor which is capable of saving the ascent profile in a "text-only" format.

The ascent profile of a balloon configuration is constructed as a number of data points which represent discrete samples from the complete curve which describes the entire vertical motion profile. Because the BDPS program will perform linear interpolation between successive points, the points should be chosen to make a linear interpolation valid. The ascent profile will consist of two columns of numbers where the first column is the time reference point and the second column is the vertical velocity corresponding to that particular time. Figure 20.2.1-1 shows the format to be followed in the creation of an ascent profile.

First Text Line:	Time 1 (s)	Vertical Velocity at Time 1 (m/s)
Second Text Line:	Time 2 (s)	Vertical Velocity at Time 2 (m/s)
(Additional Lines)		

Figure 20.2.1-1. Ascent Profile Format

20.2.2 Climate-Related Files

If the BDPS user desires to use the "climate" model option for providing wind data, then a number of files must be present within the same Macintosh folder as the BDPS application. These files are distributed on magnetic media with the BDPS application and should not be altered in content, nor should the file names be changed. Table 20.2.2-1 lists the files and gives a brief description of each file.

Table 20.2.2-1. Climate-Related Files

NAME OF DATA FILE	DESCRIPTION OF FILE CONTENTS
NASPGROVES.F	Groves data
NASPPPWCS.F	Density-velocity correlations & large scale fraction data
NASPQBO.F	Quasi-biennial oscillation data
NASPRRW.F	Random perturbation data
NASPSP.F	Spherical harmonic data
NMC.DAT	National Meteorological Center grid data
M1.DAT	Meteorological data for month of January
M2.DAT	Meteorological data for month of February
M3.DAT	Meteorological data for month of March
M4.DAT	Meteorological data for month of April
M5.DAT	Meteorological data for month of May
M6.DAT	Meteorological data for month of June
M7.DAT	Meteorological data for month of July
M8.DAT	Meteorological data for month of August
M9.DAT	Meteorological data for month of September
M10.DAT	Meteorological data for month of October
M11.DAT	Meteorological data for month of November
M12.DAT	Meteorological data for month of December

20 2.3 Wind Table Files

If the BDPS user wishes to produce a balloon drift pattern as a function of a wind model other than the GRAM climate model, the user should provide two data files: "EAST.DAT" and "NORTH.DAT." These files describe, respectively, the east and north wind velocity components as functions of latitude, longitude, and altitude. The files should be located in the same Macintosh folder as the BDPS application. Figure 20.2.3-1 shows the format for each of the two files.

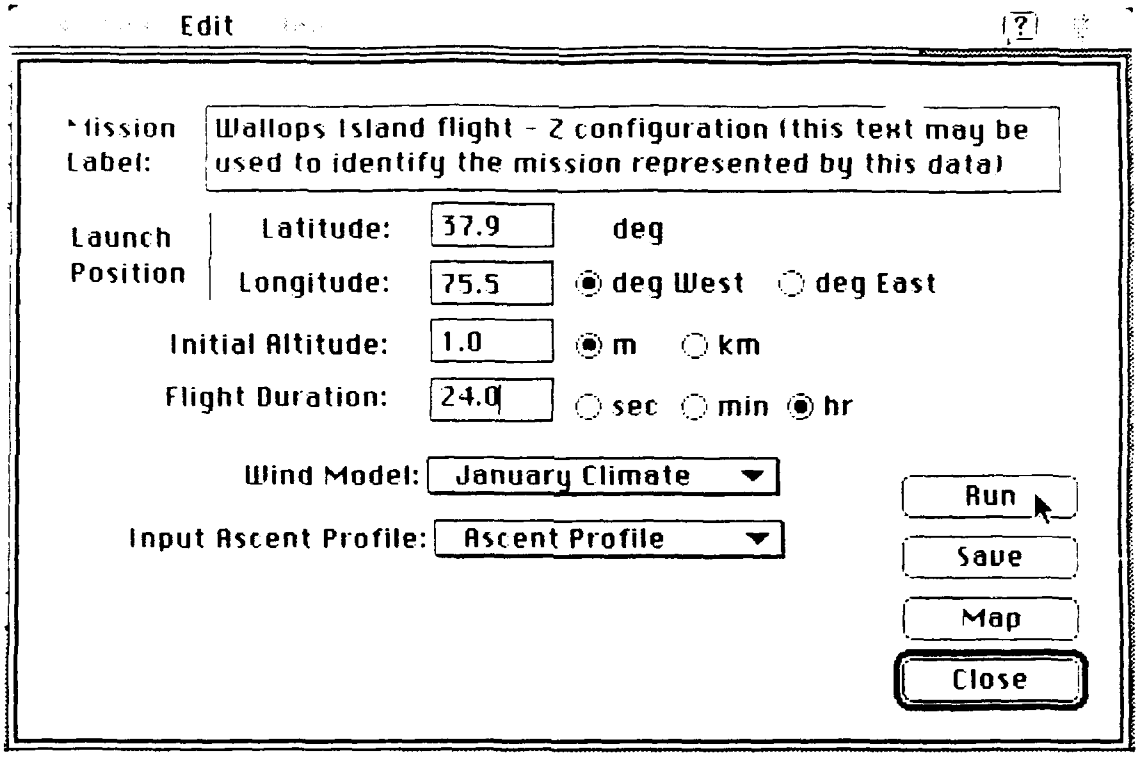
```
<no. of altitude values> <no. of longitude values> <no of latitude
     values>

<blank line>

Altitude_1    Altitude_2    ...    Altitude_k

<blank line>

Longitude_1    Longitude_2    ...    Longitude_m

<blank line>

Latitude_1    Latitude_2    ...    Latitude_n

<blank line>

<wind magnitude values:  three-dimensional nested loop with outermost
     index for latitude dependency, middle index for longitude values,
     then the innermost index for altitude values.>
```

Figure 20.2.3. Wind Table File Format

20.3 BDPS OPERATION PROCEDURE

This section describes the steps to take in the operation of BDPS to produce drift patterns. The figures in this section are screen snapshots from actual operation of the BDPS program. The procedure discussion assumes that the BDPS user already has familiarity with Macintosh procedures.

After double-clicking the BDPS application, the user must choose whether to open an existing "mission" file or to start with a new mission file. A saved mission file contains the user's run setup parameters and the data for making plots onscreen. For the sake of discussion, the user has selected "New Mission" from the File menu and will now see the run setup dialog as shown in Figure 20.3-1.



Mission Label: Wallops Island flight - 2 configuration (this text may be used to identify the mission represented by this data)

Launch Position
Latitude: 37.9 deg
Longitude: 75.5 ● deg West ○ deg East

Initial Altitude: 1.0 ● m ○ km

Flight Duration: 24.0 ○ sec ○ min ● hr

Wind Model: January Climate

Input Ascent Profile: Ascent Profile

Figure 20.3-1. Sample BDPS Run Setup Dialog

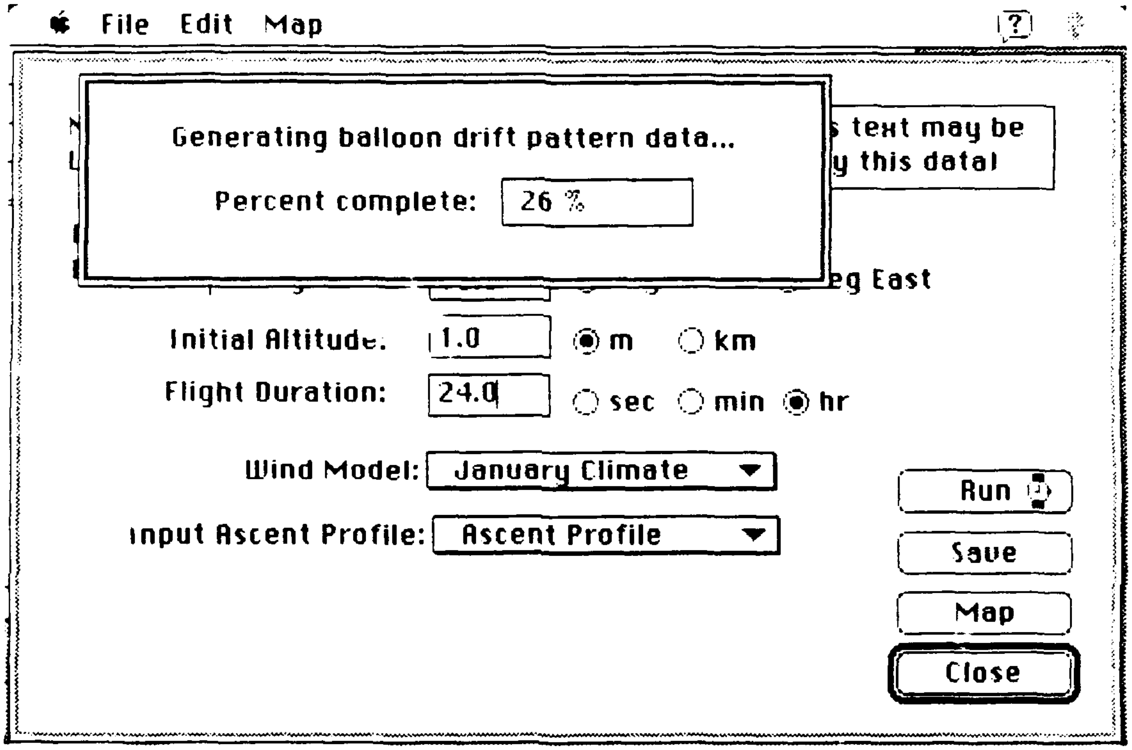
At this point, the user may enter a set of numbers in the various boxes and may select the desired units for the numbers. Table 20.3-1 lists the valid values for the various number entry boxes. If an invalid entry is made, BDPS will beep and highlight the invalid entry when the user tries to run or save with the invalid entry.

Table 20.3-1. Run Setup Ranges of Values

Entry Field	Range of Acceptable Values
Latitude	-90.0 to 90.0 degrees
Longitude	-180.0 to 180.0 degrees
Altitude	0.0 to 1,000,000.0 meters
Duration	0.0 to 720.0 hours

After entering the desired values, the user should use the four buttons in the lower-right dialog corner to take action within BDPS. The recommended sequence is top-to-bottom: run, save, map, close. When the user clicks the run button, BDPS will then generate the drift pattern data for the specified mission. During this stage, BDPS reports its progress as shown in Figure 20.3-2.

Occasionally, the status will not be updated for several moments while the climate model engages in intensive file access activity to establish or re-establish grids of data from which to interpolate.



Generating balloon drift pattern data...

Percent complete: 26 %

Initial Altitude: 1.0 ● m ○ km

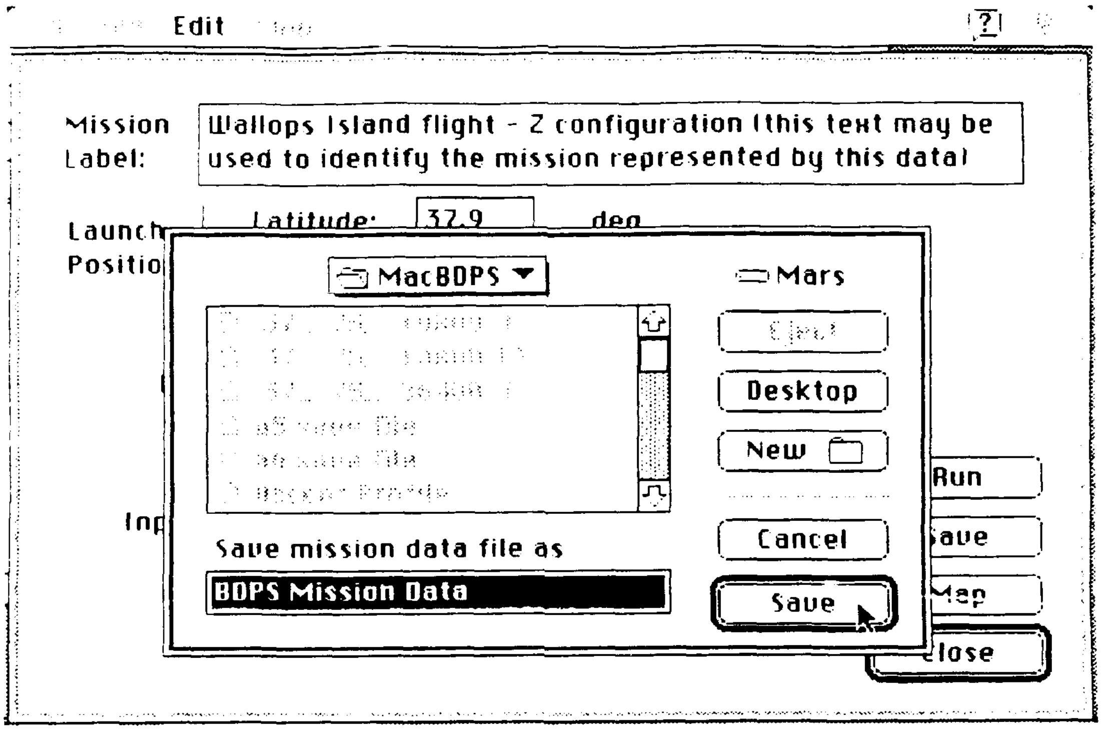
Flight Duration: 24.0 ○ sec ○ min ● hr

Wind Model: January Climate

Input Ascent Profile: Ascent Profile

Figure 20.3-2. Drift Pattern Data Generation Progress Dialog

Once the run is complete, the user should save the data which was produced. This is initiated by clicking the Save button. The user will then have the chance to specify a file name for the mission file to be saved. This operation uses the standard Macintosh interface for file operations as shown in Figure 20.3-3.



Mission Label: Wallops Island flight - Z configuration (this text may be used to identify the mission represented by this data)

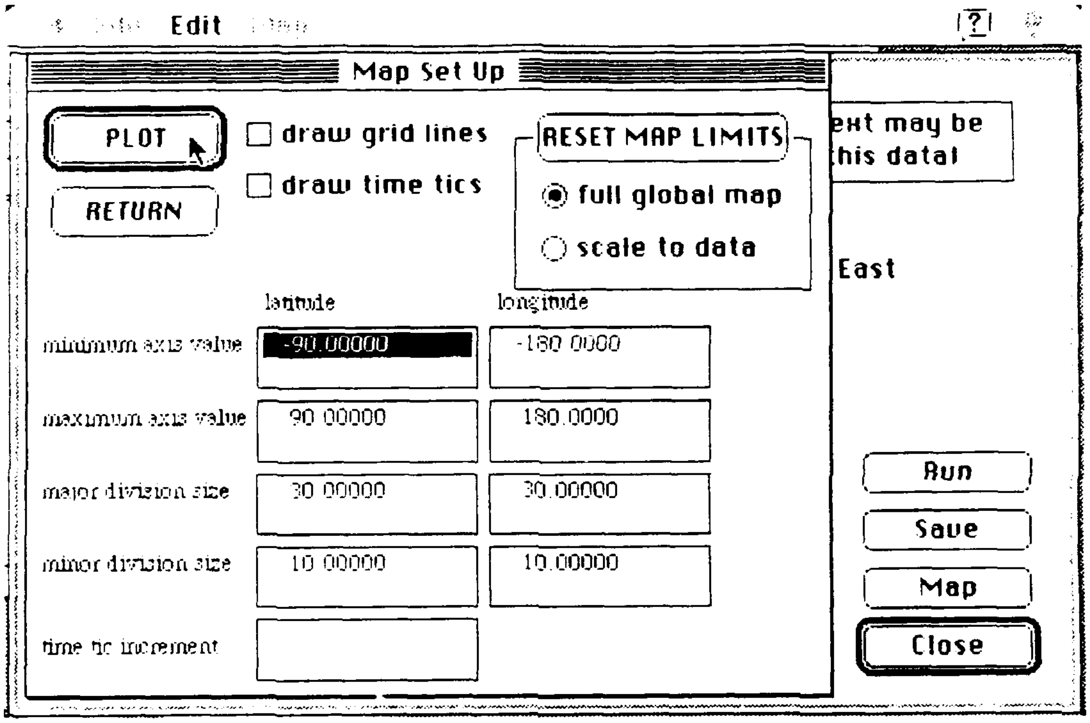
Launch Position Latitude: 37.9 deg

Save mission data file as

BDPS Mission Data

Figure 20.3-3. Saving the BDPS Mission File

Now the BDPS user may display the generated drift pattern onscreen by clicking the Map button. BDPS will present a dialog box for setting up the onscreen plot. This configuration dialog is shown in Figure 20.3-4.



Map Set Up

□ draw grid lines

□ draw time tics

● full global map

○ scale to data

	latitude	longitude
minimum axis value	-90.00000	-180.0000
maximum axis value	90.00000	180.0000
major division size	30.00000	30.00000
minor division size	10.00000	10.00000
time tic increment		

Figure 20.3-4. Setting Up the Drift Pattern Display

Once the user selects the Plot button, the screen will be cleared and the drift pattern will be displayed. BDPS produces outlines of major land masses first, and then uses a double-wide line to depict the path followed by the balloon in its trajectory. At this point, the Map menu is now active and may be used to save the map, to refresh the map, or to open a saved mission file to produce plots with previously generated data. Figure 20.3-5 shows a global map with a balloon drift originating from Wallops Island, Virginia. In that figure, the cursor is pointing to the balloon's trajectory path.

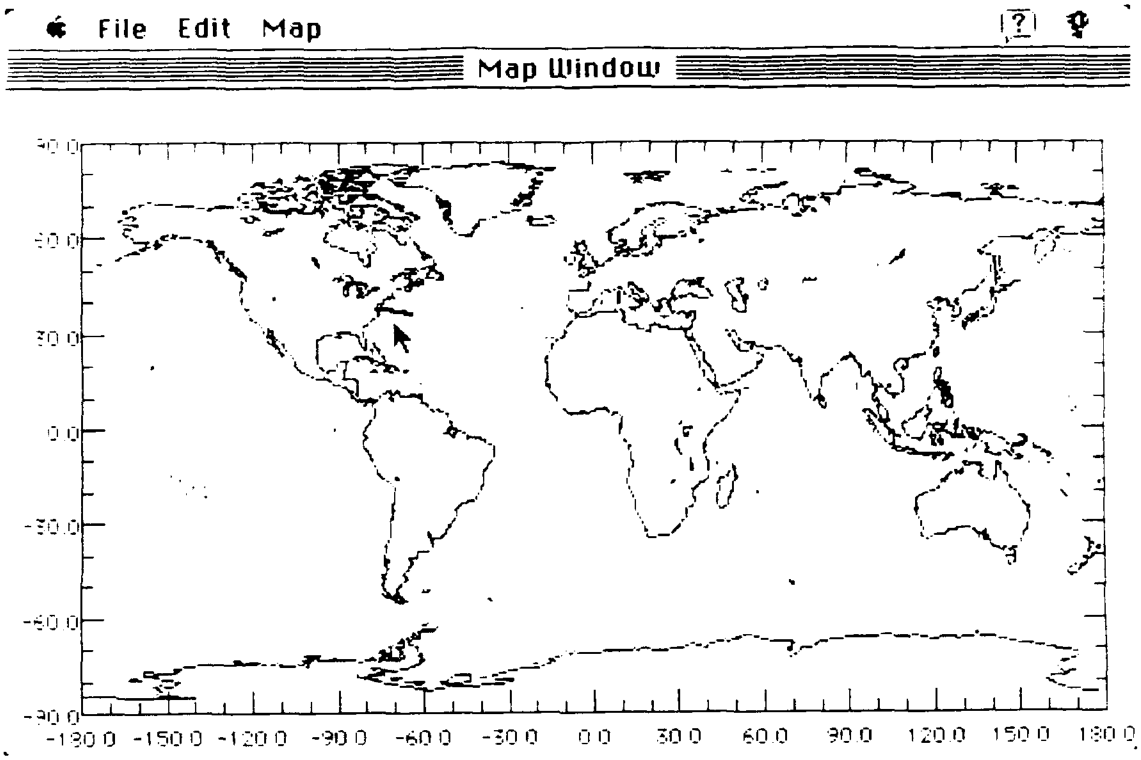
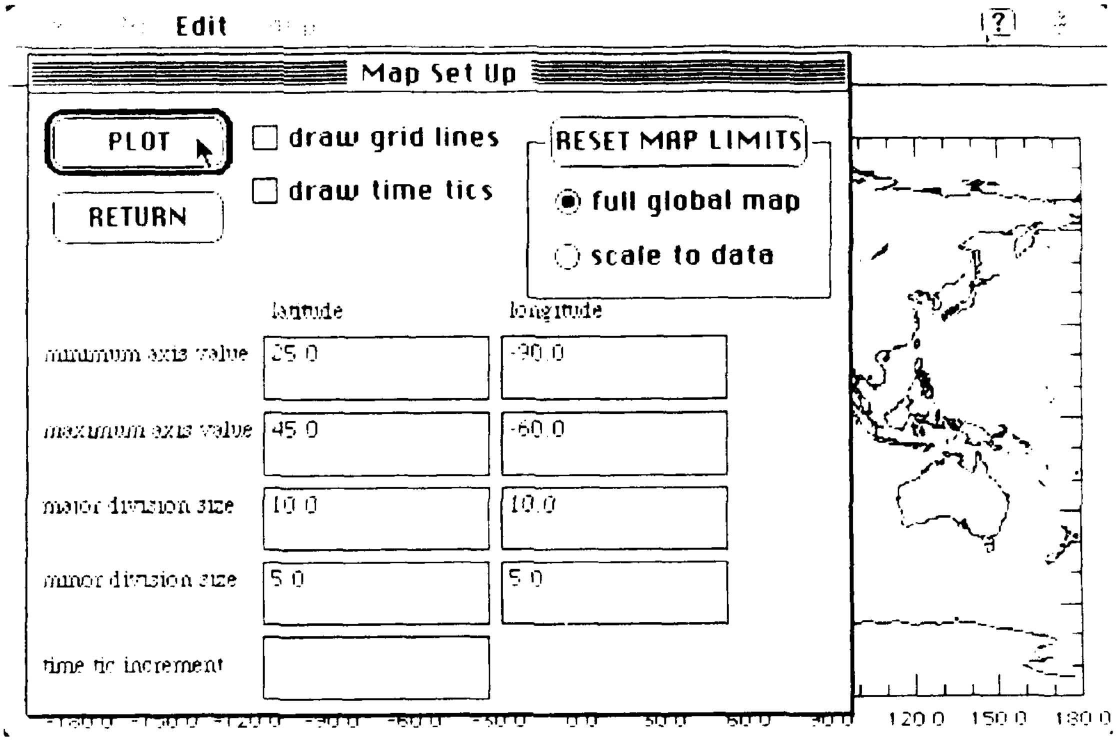


Figure 20.3-5. Sample Drift Pattern

The drift patten in the Figure 20.3-5 is a good example of a case in which the user may wish to narrow the scope of the display and "zoom in" on the region of interest. By selecting New Map from the Map menu, the user may change the map setup parameters as shown in Figure 20.3-6 and then produce a display as shown in Figure 20.3-7, which provides a clearer picture of the balloon's drift trajectory.



□ draw grid lines

□ draw time tics

● full global map

○ scale to data

	latitude	longitude
minimum axis value	25.0	-90.0
maximum axis value	45.0	-60.0
major division size	10.0	10.0
minor division size	5.0	5.0
time tic increment		

Figure 20.3-6. Providing New Map Parameters

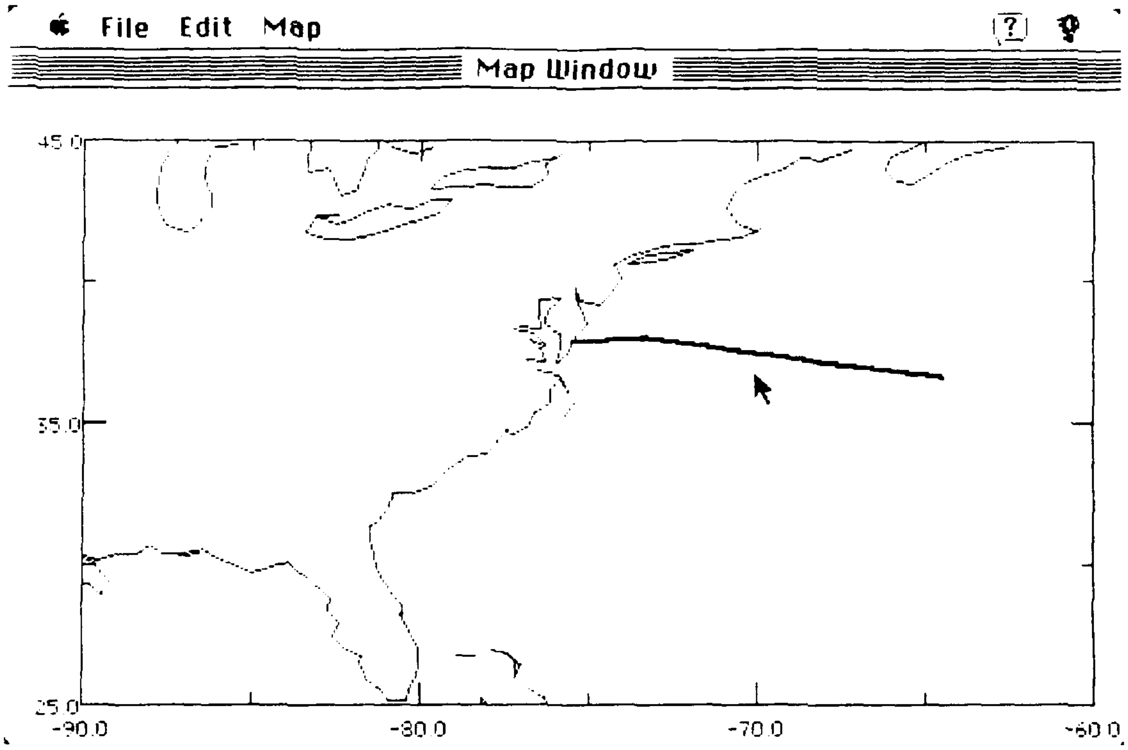


Figure 20.3-7. Sample Regional Drift Pattern

Any drift pattern plot, such as in Figure 20.3-5 or Figure 20.3-7, may be saved as a Macintosh PICT file by using the Save option from the Map menu. The user may then choose the Done option from the Map menu and the Close button from the run setup dialog. At this point, the user may Quit the application or begin with New Mission or Open Mission. After quitting BDPS, the user may start a graphics program (e.g., MacDraw II) and then open the saved PICT file which contains an image of the drift pattern display. Depending on its capabilities, the graphics program may be used to annotate and/or print the drift pattern.